Gestion optimale de l'allocation des ressources pour l'execution des processus dans le cadre du Cloud

Kahina Bessai

Département de formation doctorale en informatique

École Doctorale Management Panthéon-Sorbonne

Gestion optimale de l'allocation des ressources pour l'exécution des processus dans le cadre du *Cloud*

THÈSE

présentée et soutenue publiquement le 2 décembre 2014

pour l'obtention du

Doctorat de l'Université Paris 1 Panthéon-Sorbonne

(spécialité informatique)

par

Kahina Bessai

Composition du jury

Président :	Eric Dubois : Professeur au Centre de Recherche Public Henri Tudor, Luxembourg
Rapporteurs :	Camille Rosenthal-Sabroux : Professeur à l'Université Paris Dauphine, France Rainer Schmidt, Professeur à l'Université des Sciences Appliquées de Munich, Allemagne
Examinateur :	Claude Godart : Professeur à l'Université Lorraine, France
Directeurs de thèse :	Selmin Nurcan : Maître de conférences, HDR, à l'Université Paris 1 Panthéon-Sorbonne Colette Rolland : Professeur à l'Université Paris 1 Panthéon-Sorbonne, France

Centre de Recherche en Informatique de l'Université Paris 1 Panthéon-Sorbonne

Mis en page avec la classe thloria.

Remerciements

Je tiens d'abord à remercier Eric Dubois pour l'honneur qu'il nous a fait d'accepter de présider le jury de cette thèse.

Je tiens également à remercier Camile Rosenthal-Sabroux et Rainer Schmidt d'avoir accepté de rapporter cette thèse, de l'intérêt qu'ils ont manifesté pour notre travail ainsi que pour leurs précieux conseils pour l'amélioration de ce rapport.

Mes sincères remerciements s'adressent également à Colette Rolland pour ses conseils et ses suggestions pertinentes et efficaces qui ont considérablement contribué à l'aboutissement de ce travail. Qu'elle trouve ici tous mes sincères remerciements.

J'exprime mes remerciements et ma profonde gratitude à Selmin Nurcan sans qui cette thèse n'aurait jamais vu le jour. Je la remercie de m'avoir fait confiance, cru en moi, de m'avoir donné une chance de réaliser un rêve et surtout de m'avoir fait découvrir les processus, ces « bouts de lego avec lesquels on peut s'amuser à tout construire ». Grâce à ses idées, ses conseils, son accompagnement, sa patience et ses encouragements nous avons pu mener ce travail à bout.

Je tiens aussi à remercier chaleureusement Claude Godart pour la collaboration et l'accueil chaleureux pendant mon séjour au LORIA. Qu'il trouve ici toutes ma reconnaissance et mon profond respect.

Je remercie également Ammar Oulamara pour son accueil au LORIA et pour l'intérêt qu'il a manifesté à notre travail ainsi que pour les nombreux échanges et discussions très enrichissants et fructueux. Qu'il trouve ici tous mes sincères remerciements.

Je tiens également à remercier tous les membres et docteurs du CRI, Adrian, Elena, Bruno, Rebecca, Irina, Manuelle, Daniel, Sana, Carine, Charlotte, Hala, Camille, Raoul, Salma, Said, Kadan. Les moments partagés avec eux étaient fabuleux, des belles rencontres, des moments précieux d'amitié et de fraternité.

Le chemin parcouru pour arriver à présenter ce travail était long et pendant toutes ces années j'ai rencontré des personnes qui m'ont appris la rigueur et fait découvrir les sciences. Un grand MERCI à tous mes professeurs depuis l'école élémentaire, à tous ces forgerons du savoir et de la connaissance.

Ma gratitude et mes remerciements à ma famille. Merci à mes parents et ma grande mère de m'avoir donné envi d'apprendre, de persévérer, pour leurs sacrifices, pour leurs encouragements et soutiens sans faille. Mes rêves sont un peu les votre ...

Merci à mes frères et soeurs pour leurs soutiens surtout pendant ces derniers années qui n'ont pas été faciles pour nous ... vous êtes tous extraordinaires .

Merci à Samir de m'avoir aidé ces dernières années, d'êtres à mes cotés pour finaliser ce travail. Enfin et surtout, merci à Inès qui me comble de bonheur et ensoleille ma vie.

Á Inès

Á mes parents, frères et soeurs

Á ma grand mère Mama Gha

Á ma belle famille

Á toi Samir ...

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction générale

Ces dernières années, des investissements considérables ont été réalisés dans les technologies de l'information et de la communication (TIC). Le Web est aujourd'hui l'un des piliers de ces investissements. Son succès s'est construit autour de trois technologies : TCP-IP (*Transmission Control Protocol/Internet Protocol*), HTTP (*HyperText Transfer Protocol*) et le langage HTML (*Hypertext Markup Language*).

Utilisé initialement comme un simple moyen de partage d'information, le Web devient aujourd'hui un outil indispensable pour mieux gérer et organiser les différentes missions d'une entreprise. En effet, le contexte économique incite de plus en plus les entreprises aux fusions, acquisitions et à l'externalisation de nombreuses activités. Cette politique d'ouverture porte le nom d'*entreprise étendue*. De plus, ces dernières sont confrontées aux problèmes de communications entre les systèmes d'information distribués et hétérogènes.

Dans cet objectif, plusieurs technologies ont été proposées telles que CORBA (*Common Object Broker Architecture*), RMI (*Remote Method Invocation*) et DCOM (*Distributed Component Model*). Ces dernières se sont vite avérées insuffisantes pour gérer des systèmes d'information de plus en plus hétérogènes et ainsi une autre alternative a vu le jour. Il s'agit de l'architecture orientée service (SOA *Service Oriented Architecture*) [155][156][157]. Ceci a certainement facilité le développement d'applications distribuées et par conséquent réduit les coûts associés à leur utilisation. Cependant, les entreprises continuent d'utiliser les applications traditionnelles qui demandent des efforts considérables, souvent menés par des experts, tels que les mises à jour, la configuration, l'exécution, les tests et la sécurité des applications. De plus, ces opérations sont onéreuses. Ainsi, malgré l'introduction de ces nouvelles technologies les entreprises restent confrontées aux défis économiques qui exigent une gestion optimale des ressources informatiques

utilisées pour exécuter leurs processus métiers. C'est dans cet objectif que se développe le *Cloud Computing* [57][162][178].

Le *Cloud Computing* que l'on pourrait traduire en français par « informatique dans les nuages » est un concept récemment revenu au premier plan, mais sa première énonciation date des années soixante : « *Computation may someday be organized as a public utility*[1]. » Le terme *Cloud* est utilisé pour la première fois au début des années 90 pour désigner les réseaux informatiques fonctionnant en mode asynchrone. Quant au terme *Cloud Computing*, sa parution date de la fin des années 90 et *Salesforce*[2] fut le premier fournisseur à l'utiliser suivi de *Amazon*[3]. Les applications fournies en mode *Cloud Computing* ne se trouvent pas forcément sur un serveur hébergé par l'utilisateur mais dans un « nuage » formé de l'interconnexion de plusieurs serveurs localisés à des endroits distincts. Cette interconnexion est réalisée au niveau de fermes de serveurs appelées communément *datacenters* ou centres de traitement de données. Ceci est une conséquence directe du procédé de virtualisation. Ce dernier est le socle de ce nouveau paradigme, dont l'objectif est de faire fonctionner des applications sur différents systèmes d'exploitation sur un même serveur physique. Autrement dit, la virtualisation permet de recréer plusieurs « machines virtuelles » en utilisant une seule et même machine physique. Le *Cloud Computing* est considéré comme le nouveau paradigme de consommation des ressources informatiques. De plus en plus d'entreprises l'ont adopté pour l'exécution des applications, en raison de ses qualités avérées comme l'efficacité, la flexibilité informatique et la diminution des coûts par le partage de ressources (stockage et/ou calcul).

Bien qu'il n'existe pas encore de définition officielle du *Cloud Computing*, celle du *National Institute of Standards and Technology (NIST)*[4] est plus ou moins devenue la norme de fait : *« Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. »* Cette définition traite des éléments suivants, détaillés dans le chapitre suivant : (1) un accès en libre-service à la demande, (2) un accès ubiquitaire au réseau, (3) une mise en commun des ressources, (4) une « élasticité rapide » et (5) un service mesuré en permanence. D'un point de vue commercial, le

1. Enoncé en 1960 par John McCarthy
2. http://www.salesforce.com
3. http://aws.amazon.com/
4. http://www.nist.gov/index.html

modèle économique du *Cloud Computing* est basé sur *pay-as-you-go*[5]. Il est constitué de trois types de services (applications[6], plateformes[7] et infrastructures[8]) et de trois types d'usage (public, privé et hybride). Généralement, trois types d'acteurs sont distingués : le fournisseur de la plateforme matérielle du *Cloud*, le fournisseur de services logiciels et le consommateur des services déployés.

Comme le *Cloud Computing* couplé avec l'architecture orientée service semble être la solution la plus intéressante et la plus élaborée pour l'exécution des processus, au cours de ces dernières années, la migration des applications d'entreprises vers le *Cloud Computing* n'a cessé de s'accentuer. Ce nouveau paradigme a ainsi profondément modifié le mode de consommation des services informatiques (matériel et logiciel). En effet, son modèle économique propose une transition d'un modèle dans lequel les utilisateurs se chargent du maintien des logiciels et des infrastructures vers un modèle dans lequel les utilisateurs sont des loueurs de services, qu'ils soient logiciels ou matériels. Le *Cloud Computing* dématérialise ainsi l'espace physique dans lequel les données sont gérées et les calculs sont effectués.

De ce fait, de nombreuses entreprises sont séduites par cette promesse d'utiliser des ressources disponibles de manière efficace, presque sans limite, tout en minimisant le temps d'exécution des applications et le budget induit par l'utilisation d'un ensemble de ressources. Dans ce sens, le travail présenté dans cette thèse consiste à proposer un ensemble de méthodes qui permettent d'utiliser une infrastructure du type Cloud Computing d'une manière optimale dans la gestion des processus.

En effet, tandis qu'un engouement majeur et incontestable soutient le développement du *Cloud Computing*, des doutes liés aux questions sécuritaires et de qualité de service peuvent freiner son essor et même compromettre son avenir. De plus, les utilisateurs sont confrontés à un problème majeur qui est le manque d'outils les aidant à choisir les meilleurs fournisseurs pour exécuter leurs processus métier. C'est dans cette optique que s'inscrivent les travaux présentés dans ce manuscrit. Plus précisément, nous nous sommes intéressés au problème du choix des meilleurs services et fournisseurs à invoquer pour exécuter un processus métier donné, tout en respectant les contraintes organisationnelles, fonctionnelles, non fonctionnelles et techniques qui les régissent. Autrement dit, notre objectif est d'ordonnancer les tâches d'un processus métier en utilisant différents fournisseurs

5. « payer en fonction de la consommation réelle »
6. SaaS : Software as a service
7. PaaS : Platform as a service
8. IaaS : Infrastructure as a service

de services.

L'ordonnancement est un aspect fondamental dans le cadre du *Cloud Computing* car il permet la parallélisation des applications et l'amélioration de leurs performances [150]. L'ordonnancement d'un processus consiste à respecter les contraintes (en particulier les contraintes de précédence des tâches) qui lient les différentes tâches le constituant et à déterminer les ressources à leur affecter. Autrement dit, un problème d'ordonnancement consiste à organiser l'exécution d'un ensemble de tâches liées par des contraintes de précédence, de données et de ressources.

Par conséquent, les algorithmes d'ordonnancement dépendent aussi bien de l'architecture cible (liens entre les ressources) que de la structure du modèle de processus à exécuter. Les processus auxquels nous nous sommes intéressés, dans le cadre de cette thèse, sont de deux types [124] : (1) les processus scientifiques et (2) les processus métiers modélisés sous forme d'un graphe sans cycle (*DAG, Directed Acyclic Graph*) où les sommets représentent les tâches et les arcs représentent les contraintes de précédence.

Le problème d'ordonnancement des tâches d'un graphe sans cycle a été largement étudié lorsque l'on utilise des ressources hétérogènes déployées dans des grilles de calcul [77][78][79]. Étant donné l'appartenance du problème d'ordonnancement de graphes de tâches sans cycle à la classe NP-complet, les algorithmes proposés sont plutôt des heuristiques. Ces dernières peuvent être scindées en deux catégories principales, à savoir :

1. Les heuristiques d'ordonnancement dites de *liste*, qui se déroulent en deux phases. D'abord une liste de tâches est constituée en attribuant une priorité à chacune des tâches la constituant. Ces priorités sont calculées en respectant les contraintes de précédence. Ces tâches sont ensuite sélectionnées dans l'ordre de leurs priorités et chaque tâche sélectionnée est ordonnancée sur la ressource qui minimise une fonction objective (généralement, la date au plus tard de sa fin).
2. Les heuristiques dites de *clustering* qui exploitent l'idée d'affecter les tâches à des groupes (*clusters*) de façon à ce que les tâches appartenant à un même groupe puissent être exécutées en parallèle.

Cependant, ces approches ne peuvent pas être appliquées au problème auquel nous nous sommes intéressés et souffrent de plusieurs insuffisances dont les plus significatives se déclinent selon les points suivants :

1. Elles ne prennent en compte qu'un seul critère d'optimisation, à savoir le temps d'exécution (appelé *makespan*). Or, le modèle économique du *Cloud Computing*,

comme mentionné précédemment, est basé sur la consommation réelle des utilisateurs. Il serait donc plus judicieux de prendre en compte un deuxième critère d'optimisation : le coût engendré par l'utilisation d'un ensemble de ressources.

2. Elles transforment le problème initial, l'ordonnancement d'un graphe de tâches, en utilisant des fonctions d'agrégation (moyenne, maximum, minimum) pour représenter le temps de transfert des données entre les différentes tâches.
3. Elles supposent que les ressources mises à la disposition des utilisateurs sont finies. Or, une des caractéristiques essentielles du *Cloud Computing* est la possibilité d'utiliser autant de ressources que nécessaire pour l'exécution d'une application donnée. Il serait donc plus approprié de supposer que les ressources mises à la disposition des utilisateurs par des fournisseurs du *Cloud Computing* sont en nombre infini.

Pour pallier ces insuffisances, nous proposons dans cette thèse des approches plus appropriées pour l'allocation des ressources et l'ordonnancement des processus en prenant en compte deux critères conflictuels, à savoir le temps d'exécution et le coût engendré par l'utilisation d'un ensemble de ressources. De plus, lorsque plusieurs instances d'un processus métier accèdent de manière concurrente aux ressources disponibles un troisième critère peut être pris en compte. Il s'agit du critère de l'« équité », permettant comme son nom l'indique d'assurer un partage équitable des ressources entre les différentes instances du processus.

En résumé, les problèmes à résoudre et les aspects auxquels nous nous sommes intéressés dans cette thèse sont multiples. Le premier aspect consiste à élaborer un modèle d'allocation de ressources et d'ordonnancement de tâches des processus qui prenne en compte les spécificités du *Cloud Computing*, à savoir l'évolutivité, l'élasticité ainsi que sa dimension économique. Le deuxième aspect consiste à définir des approches efficaces, au sens de la complexité temporelle, pour l'utilisation des ressources tout en minimisant deux critères des plus importants lors de l'exécution d'un processus donné (le temps global d'exécution et le coût engendré par l'utilisation d'un ensemble de ressources). Finalement, il faut prendre en compte le fait que toutes les tâches d'un processus ne sont pas automatisées d'une part, et d'autre part, que plusieurs instances d'un même processus peuvent être amenées à solliciter les ressources nécessaires de manière concurrente. Le cadre de l'analyse de notre travail est celui des graphes de tâches sans cycle et la technique utilisée est celle de l'optimisation multi-objectifs. Plus précisément, notre objectif est de proposer des algorithmes efficaces pour l'exécution des processus, modélisés sous forme de graphes sans cycle, dans le contexte du *Cloud computing*, tout en prenant en compte deux critères

de qualité de service des plus importants, à savoir le temps d'exécution et le coût engendré par l'utilisation d'un ensemble de ressources.

Nos contributions se déclinent comme suit :

1. La proposition de trois approches complémentaires pour une exécution optimisée d'un *workflow scientifique* en prenant en compte le temps d'exécution et le coût engendré par l'utilisation d'un ensemble de ressources.
2. L'extension des approches proposées pour l'exécution des workflows scientifiques, lorsque l'on considère une *instance* d'un processus métier. Dans cette partie, nous avons considéré le cas où les tâches constituant un processus ne sont pas toutes entièrement automatisées. Autrement dit, l'intervention de l'humain est nécessaire pour l'exécution de certaines tâches.
3. La proposition d'approches pour l'ordonnancement et l'allocation des tâches d'un *processus métier* en prenant en compte l'accès concurrent aux ressources des *différentes instances* d'un processus métier donné en considérant en particulier les patrons de flux de contrôle.

Organisation du rapport

Le présent rapport est organisé en deux grandes parties. La première partie est consacrée à la définition de la problématique à laquelle nous nous sommes intéressés et décrit brièvement nos contributions. Elle propose aussi un survol des notions de base nécessaires à la compréhension de notre travail ainsi qu'une étude bibliographique des travaux annexes traitant de la problématique d'allocation de ressources et d'ordonnancement de tâches des processus. Elle souligne les limites des travaux existants et détaille les apports de notre travail par rapport à ces derniers.

La deuxième partie constitue le cœur de notre travail. Elle présente les solutions que nous avons proposées pour traiter la problématique posée.

L'organisation de ce rapport suit le schéma décrit à la la Figure 1.1 :

Chapitre 2

Ce chapitre présente le contexte général de notre travail, à savoir l'allocation des ressources et l'ordonnancement des tâches des processus et les principales motivations de notre travail ainsi qu'un résumé de nos contributions.

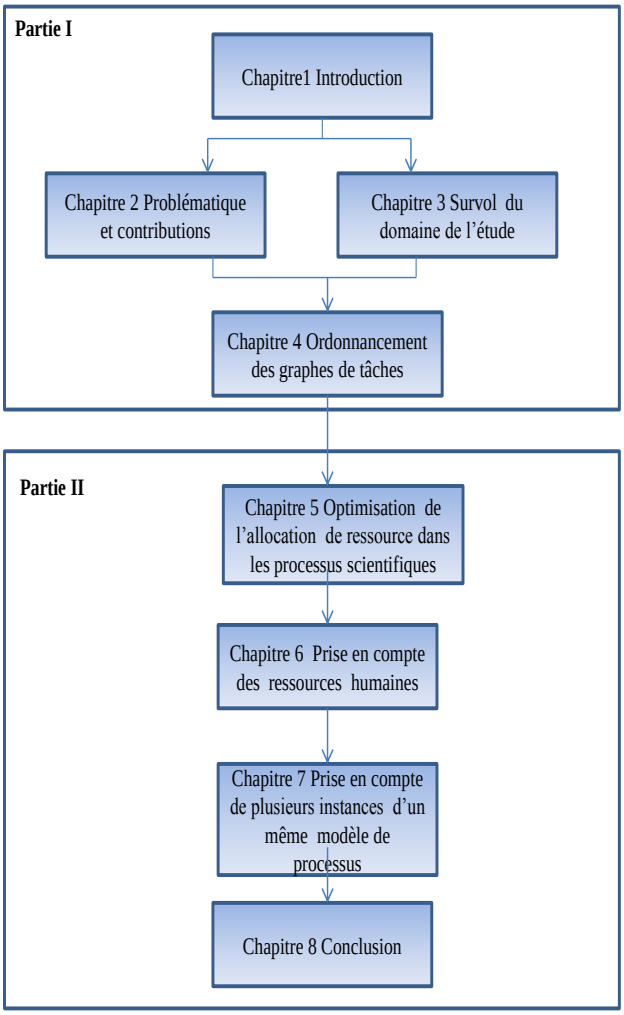

FIGURE 1.1 – Structure et chapitres de la thèse

Chapitre 3

Ce chapitre est consacré à une brève présentation de l'informatique dans les nuages (*Cloud Computing*) ainsi que des deux types de processus concernés par notre travail, à savoir : les processus scientifiques et les processus métiers. La différence essentielle, de point de vue du travail présenté dans ce rapport, est que les tâches constituant le premier type de processus sont toutes entièrement automatisées (elles sont par conséquent exécutées sur des machines virtuelles) et que dans le deuxième type de processus, certaines tâches du processus nécessitent l'intervention de ressources humaines pour les exécuter. D'autre part, lorsque l'on considère les processus métiers, plusieurs instances du même modèle de processus sont amenées à partager des ressources ou à accéder de manière

concurrente aux ressources disponibles.

Chapitre 4

Ce chapitre est consacrée aux travaux annexes au travail présenté dans ce manuscrit. Nous commençons par introduire le problème d'allocation de ressources et d'ordonnancement des graphes de tâches, puis nous exposons les différentes approches candidates pour le résolution du problème auquel nous nous sommes intéressés.

Chapitre 5

L'objectif de ce chapitre est de présenter des approches d'allocation de ressources et d'ordonnancement de tâches dans un processus scientifique. Les contributions de ce chapitre se déclinent en quatre points :

1. Modélisation du problème de l'allocation de ressources et d'ordonnancement des processus scientifiques sur une plate-forme de *Cloud Computing*.
2. Proposition d'une première approche basée sur la minimisation du coût d'exécution d'un processus scientifique engendré par l'utilisation d'un ensemble de ressources dans le cloud (machines virtuelles).
3. Proposition d'une deuxième approche basée sur la minimisation du temps d'exécution d'un processus scientifique.
4. Proposition d'une troisième approche combinant les deux premières approches, en prenant en compte les deux objectifs simultanément.

Chapitre 6

Ce chapitre est consacré à l'extension des travaux présentés dans le chapitre précédent. Plus précisément, l'objectif est de prendre en compte le fait que lorsque l'on s'intéresse à une instance d'un processus métier toutes les tâches constituant le modèle de processus ne sont pas entièrement automatisées. En d'autres termes, l'intervention de ressources humaines est indispensable pour accomplir certaines (voire toutes les) tâches du processus métier.

Chapitre 7

Nous présentons dans ce chapitre un ensemble de méthodes pour l'allocation de ressources et l'ordonnancement des tâches pour toutes les instances d'un modèle de processus métier. En plus des deux critères d'optimisation pris en compte précédemment (le temps et le coût d'exécution), un troisième objectif est considéré : il s'agit de l'équité. Une mesure permettant d'assurer un partage équitable des ressources lorsque plusieurs instances de processus les invoquent de manière concurrente. Le problème majeur dans ce type d'ordonnancement est la gestion de l'équité qui consiste à trouver des ordonnancements tout en prenant en compte la sollicitation concurrente des ressources humaines par plusieurs instances du modèle de processus.

Chapitre 8

Ce chapitre résume nos contributions et dégage des perspectives à court et à long terme pour nos travaux de recherche.

Une bibliographie guidera le lecteur tout au long de sa lecture.

Chapitre 2

Problématique & contributions

Sommaire

2.1 Introduction

Le *Cloud Computing* promet une révolution du monde de l'IT en proposant des services à la demande (de calcul et/ou de stockage de données) accessibles via Internet. Un des points essentiels de ce nouveau paradigme est la notion d'extensibilité à la demande et d'élasticité. Autrement dit, les utilisateurs ne paient que leur consommation réelle. C'est l'un des avantages majeurs par rapport à une infrastructure propre à une entreprise où les ressources informatiques sont très souvent sous-utilisées. Pour résumer, le *Cloud Computing* c'est :

a. L'abstraction sur la localisation des ressources utilisées.

b. Le partage des ressources disponibles.

c. L'ajout des ressources au système à la demande. Rappelons que ces ressources sont considérées être en nombre « infini ».

d. La facturation à la demande.

e. Le self-service via Internet.

De ce fait, quasiment toutes les entreprises soucieuses d'une bonne gestion de leurs ressources informatiques ont eu recours au *Cloud Computing*. Avant l'émergence de ce nouveau paradigme, la solution prédominante était d'utiliser des ressources locales avec un grand nombre de paramètres à gérer tels que les mises à jour des logiciels utilisés, le refroidissement des serveurs, etc. Ainsi, le passage à ce nouveau mode de consommation des ressources a considérablement réduit les coûts induits par leur utilisation.

Cependant, malgré les nombreux avantages avérés d'utiliser des ressources déployées sur le nuage, le *Cloud Computing* reste confronté à de nombreux problèmes dont le manque de méthodes et d'outils aidant les utilisateurs à mieux gérer les ressources mises à leur disposition. En effet, on trouve de plus en plus de fournisseurs de services qui proposent des services avec les mêmes fonctionnalités. La différence entre ces offres est au niveau de la qualité de service offerte. Autrement dit, les utilisateurs des ressources du *Cloud Computing* sont confrontés au problème du choix du meilleur fournisseur de service pour la mise en oeuvre des fonctionnalités de leurs applications. Étant donné un ensemble de services offrant les mêmes fonctionnalités, les questions auxquelles il faut répondre peuvent être donc formulées comme suit :

a. Quels sont les meilleurs fournisseurs à recommander à un utilisateur de façon à le satisfaire au mieux ?

b. Quels sont les critères de qualité de service à prendre en compte ?

c. Quelle est l'approche la plus appropriée pour répondre à ces questions ?

C'est dans cette optique que s'inscrivent les travaux présentés dans cette thèse. Plus précisément, nous nous sommes intéressés au problème du choix des meilleurs ressources à utiliser pour l'exécution d'un processus donné, de façon à minimiser deux critères de qualité de service, à savoir le temps et le coût d'exécution. Avant de donner plus de détails sur les approches proposées pour une gestion optimale des ressources lors de l'exécution des processus (scientifiques et métiers), nous commençons, dans ce chapitre, par situer le contexte général de nos travaux et la problématique à laquelle nous nous sommes intéressés.

La suite de ce chapitre est organisée comme suit. La prochaine section expose le contexte général de nos travaux. La Section 2.3 décrit la problématique traitée dans cette thèse et les questions de recherche auxquelles nous nous sommes intéressées. La Section 2.4 rapporte nos différentes contributions. Ces dernières peuvent être scindées en trois parties complémentaires. La première partie consiste à proposer un ensemble de méthodes pour l'allocation des ressources et l'ordonnancement des tâches dans des processus scientifiques. La deuxième contribution consiste à prendre en compte le fait que pour certaines tâches, constituant une instance d'un processus donné, l'intervention des ressources humaines est nécessaire pour leur réalisation. La troisième et dernière contribution permet de prendre en compte et d'administrer l'exécution, de manière concurrente, de plusieurs instances d'un même modèle de processus métier. La Section 2.5 conclut ce chapitre.

2.2 Contexte de la thèse

Ces dernières années, la familiarisation avec l'utilisation de nouvelles technologies de l'information et de la communication a fait apparaître de nouvelles exigences et attentes de la part des acteurs métiers du manque économique. Dans la perspective de faire face à cela, les entreprises se sont tournées vers l'externalisation de certaines de leurs activités. Le but est de produire plus vite et à moindre coût. L'architecture orientée service a pu être mise en oeuvre dans ce contexte et a apporté des réponses méthodologiques et techniques à ces exigences. L'architecture orientée service permet de coupler des systèmes d'information hétérogènes et de réutiliser les services disponibles. Ce type d'architecture a considérablement contribué à la diminution des coûts liés à l'utilisation des applications accessibles via Internet. Cependant, les entreprises continuent d'utiliser le modèle

traditionnel d'acquisition de licences logicielles déployées sur leurs propres serveurs. Ces applications sont souvent onéreuses et exigent des compétences approfondies pour assurer de nombreuses opérations telles que : (i) l'installation, (ii) les mises à jour, (iii) les tests et (iv) l'exécution, engendrant ainsi de nombreuses difficultés. Pour y remédier, un nouveau paradigme a vu le jour s'opposant au modèle traditionnel d'acquisition de logiciels dans le sens où il permet aux entreprises qui l'adoptent de ne presque plus gérer aucun matériel ni logiciel. Tout cela devient alors du ressort d'un (ou de plusieurs) fournisseur(s) de services. En effet, à titre d'exemple, en mode SaaS (*Software as a Service*) [9], les utilisateurs accèdent à leur messagerie depuis n'importe quelle machine via Internet en se connectant à l'infrastructure du prestataire. La gestion de cette infrastructure n'est donc plus du ressort de ses utilisateurs (ni de leur entreprise). L'exploitation côté utilisateurs se réduit à la gestion des comptes via des outils fournis par le prestataire. Plus précisément, ce nouveau paradigme se réfère à la mise à disposition, des utilisateurs, des services à la demande et extensibles à volonté. En contradiction avec les systèmes actuels, les services mis à la disposition d'une entreprise sont virtuels et illimités en nombre. De plus, les détails des infrastructures physiques, utilisées pour déployer ces services, ne sont plus du ressort des entreprises clientes. Le *Cloud* est ainsi divisé en trois couches :

1. Applicative (*SaaS, Software as a Service*) « le logiciel en tant que service » repose sur la mise à disposition, à travers Internet, des applications consommées à la demande et en fonction de l'utilisation réelle.
2. Plateforme (*PaaS, Platform as a Service*) « la plateforme en tant que service » est la plateforme utilisée pour le déploiement, le développement et l'exécution des applications utilisées.
3. Infrastructure (*IaaS, Infrastructure as a Service*) « l'infrastructure en tant que service » est l'ensemble du matériels utilisés pour le déploiement et l'exécution des applications en question.

Il existe aussi trois types de *Cloud* :

1. Les *Cloud* privés dont deux types sont à distinguer, à savoir : (1) le *Cloud* privé interne et le *Cloud* privé externe. Mais généralement lorsque l'on parle de *Cloud* privés il s'agit souvent de *Cloud* privés externes dédiés à une entreprise dont la gestion est externalisée à un prestataire.

9. Par exemple Google Apps

2. Les *Cloud* publiques se réfèrent à un service utilisé par une entreprise et géré par un tiers.
3. Les *Cloud* hybrides sont une combinaison des deux premiers types de *Cloud*.

Le socle de ce nouveau paradigme est la virtualisation. C'est une technique utilisée pour une gestion optimisée des ressources matérielles. Plus précisément, elle permet d'instancier plusieurs machines virtuelles sur une même machine physique et d'éviter ainsi la sous-consommation des ressources. En effet, la plupart des serveurs informatiques sont rarement utilisés à leur capacité maximale. La technique de virtualisation permet ainsi d'être plus flexible dans l'allocation des ressources utilisées. Effectivement, si par exemple une machine virtuelle manque de ressources sur un serveur car ce dernier est à sa capacité maximale, alors cette machine virtuelle est instanciée sur un autre serveur.

Coupler le paradigme du Cloud avec l'architecture orientée service est certainement la solution la plus appropriée pour une meilleure gestion des ressources de calcul et/ou de stockage de données et l'exécution des processus d'entreprise [133][134]. Ce couple est, en effet, utilisé par de nombreuses entreprises pour exécuter leurs applications. Ils (Cloud et SOA) ont permis à la technologie de l'information de réduire les coûts des transactions.

Par ailleurs, les systèmes de gestion de workflows (Wfms) permettent la gestion du flux de travail dans les organisations. Initialement développés comme outils de *groupware* et généralisés pour la gestion des processus d'entreprise et ils sont aussi utilisés, par exemple, dans le cadre de l'exécution des processus scientifiques dans de nombreux domaines tels que (i) l'astronomie, (i) la géologie et (iii) la bio-informatique [52]. Ces processus qui sont gérés par des systèmes de gestion de workflows sont composés d'un ensemble de tâches à exécuter suivant un flux de travail (c'est-à-dire, les tâches respectent des contraintes de précédence). Le Wfms répondent a plusieurs besoin concernât ces processus. Nous distinguons généralement deux types de besoin : (i) le besoin d'exécution (*performing*) pour les tâches automatiques et (ii) le besoin de stockage pour sauvegarder les données nécessaires à l'exécution des tâches constituant le workflow en question. Un troisième besoin s'ajoute aux deux premiers pour les processus métiers, capacité à faire intervenir des ressources humaines pour l'exécution des processus.

Le *Cloud* répond aux deux besoins cités précédemment, à savoir le stockage (ou sauvegarde) de données et l'exécution des tâches automatiques. En effet, plusieurs études ont montré l'intérêt d'utiliser une infrastructure de type *Cloud* en la comparant notamment avec les solutions existantes telles que le déploiement en local et l'utilisation d'une infrastructure de type grilles de calcul (*Grid Computing*). Les raisons de ce constat sont

multiples dont les plus significatives se déclinent comme suit :

a. réduction des coûts induits par l'utilisation des ressources requises pour l'exécution d'un processus ;

b. recherche d'une meilleure performance notamment pour assurer un seuil de qualité de service donné ;

c. recherche d'une meilleure capacité à monter en charge.

Il existe des algorithmes efficaces dans la littérature pour l'ordonnancement et l'allocation des tâches d'un workflow scientifique, tels que ceux proposés dans le cadre de l'utilisation d'infrastructures de type grilles de calcul. Ces algorithmes restent cependant inappropriés pour une infrastructure de type *Cloud*. En effet, contrairement aux grilles de calcul où le nombre de ressources mises à la disposition des utilisateurs est fini, dans le cadre du *cloud* le nombre de ressources est, en principe, infini [125][126][127]. Ainsi, un utilisateur peut utiliser autant de ressources que nécessaire à l'exécution de ses applications. Cette caractéristique est l'une des plus importantes du paradigme *Cloud computing*. Elle est appelée « illusion of infinite resources ». Une autre caractéristique importante du *Cloud* est son élasticité qui permet de fournir des services évolutifs et par conséquent de supporter les montées en charge. Autrement dit, l'élasticité du paradigme *Cloud computing* permet aux utilisateurs de louer et de libérer des ressources de calcul et/ou de sauvegarde des données en fonction de leurs besoins. En effet, et à titre d'exemple, Salesforce (`http:///www.salesforce.com`) - l'un des pionniers dans le domaine de l'informatique dans le nuage, gère les données de plus de 54 000 entreprises, avec seulement 1 000 serveurs (mars 2009).

C'est dans ce contexte d'optimisation des ressources déployées dans le cadre du *Cloud* que s'inscrivent les travaux présentés dans cette thèse.

2.3 Problématique de la thèse

L'un des premiers avantages de migrer des processus d'entreprise vers le *Cloud Computing* est d'assurer *la montée en charge* en temps réel. Effectivement, contrairement par exemple aux grilles de calcul, l'approvisionnement en ressources dans le cadre du *Cloud Computing* permet la variation du type et de la quantité des ressources à utiliser pour l'exécution des processus en question. Autrement dit, lorsque l'on utilise des ressources déployées dans le *Cloud Computing*, il est possible d'augmenter ou de diminuer des ressources en fonction des besoins des processus à exécuter. Par conséquent, ce paradigme

permet aux systèmes de gestion de workflows de satisfaire au mieux les clients en assurant ainsi un niveau de qualité de service que les fournisseurs s'engagent à respecter. En effet, dans le but d'assurer un niveau de qualité de service, un contrat de service (SLA, *Service Level Agreement*) [10][158] [159] [160] [161] est généralement négocié entre le fournisseur de services et le client.

Les propriétés d'élasticité et d'illusion de ressources infinies qu'offre le *Cloud computing* sont deux des aspects les plus prometteurs de ce paradigme. Elles permettent aux entreprises de louer et de libérer des ressources en fonction des besoins de leurs applications. Ceci, sans se soucier ni de la gestion de l'infrastructure utilisée ni de la mise à jour, par exemple, des outils utilisés.

Cependant, malgré les nombreux avantages prouvés d'utiliser une infrastructure du type *Cloud computing*, l'utilisateur reste confronté à de nombreux problèmes majeurs qui peuvent compromettre le succès de ce type de solution, par exemple la difficulté du choix du meilleur fournisseur parmi un ensemble de fournisseurs possibles [128][129][130][131][132][135]. Cette difficulté s'accroit lorsque l'on considère plusieurs critères de qualité de service souvent conflictuels. De plus, pour exécuter une tâche donnée d'un processus plusieurs offres sont disponibles et de plus en plus nombreuses. En effet, avec l'émergence de la technologie *Cloud*, tous les « géants » de l'informatique tels que Amazon, Microsoft, IBM, ... se sont mis à proposer des offres diverses et variées. Les offres disponibles sur le marché ont des modèles économiques (*Business Model*) différents. D'autre part, l'utilisation de ressources dans le Cloud telles que les machines virtuelles engendrent de nouveaux coûts tels que le coût d'exécution, le coût de transfert, le coût d'ouverture d'une nouvelle machine virtuelle dont le mode de calcul dépend du modèle économique utilisé par le fournisseur de service sollicité. Les temps d'exécution aussi sont différents selon les performances des ressources choisies. Tous ces paramètres nous conduisent à la question de recherche à laquelle nous nous sommes intéressées, à savoir :

« Quelle est la meilleure sélection ou composition de ressources (machines virtuelles et humaines) pour l'exécution d'un processus dans le Cloud ? »

En outre, nous avons constaté qu'il existe des besoins différents selon le type de processus considéré : (1) *processus scientifique* et (2) *processus métier*. En effet, lorsque l'on exécute le premier type de processus, généralement une seule instance est considérée à la fois et toutes les tâches sont automatisées. Par contre, lorsque le deuxième type de

10. Le SLA est un document qui définit la qualité de service requise entre le fournisseur d'un service et son(ses) utilisateur(s)

processus est considéré, il est rare d'en exécuter qu'une seule instance et d'automatiser toutes les tâches qui le composent. Autrement dit, il convient pour ce type de processus de distinguer deux types de ressources : (1) ressources humaines et (2) ressources informatiques (machines virtuelles). D'autre part, il convient d'étudier le cas où plusieurs instances accèdent de manière concurrente à un ensemble de ressources.

Dans cette thèse, nous nous sommes donc intéressés au guidage des choix des ressources à utiliser pour l'exécution des workflows, tout en prenant en compte deux critères de qualité de service : (i) le temps d'exécution et (ii) le coût engendré par l'utilisation d'un ensemble de ressources. Le cadre de l'analyse est celui des workflows et l'approche utilisée est celle de l'*optimisation bi-objectives*. Ce problème n'a pas laissé la communauté des grilles de calcul indifférente et plusieurs solutions ont été proposées notamment pour la minimisation du temps d'exécution d'un processus donné. Cependant, ces approches s'appliquent à un nombre borné de ressources. Elles ne sont donc pas optimales dans le cadre du *Cloud* pour les raisons mentionnées précédemment. Il est semble alors plus judicieux : (i) de supposer que le nombre de ressources mises à la disposition des utilisateurs sont infinies et (ii) de prendre en compte, en plus du temps d'exécution, le critère du coût d'exécution étant donné que le modèle économique du *Cloud computing* est basé sur « payer en fonction de la consommation réelle ». Par ailleurs, le problème d'ordonnancement des tâches liées par des contraintes de précédence, que ce soit sous sa forme générale ou dans de nombreux cas particuliers, appartient à la classe de problèmes NP-complet. Il est plus raisonnable dans ce cas de proposer des algorithmes (heuristiques) efficaces que de proposer des algorithmes exacts.

Nos travaux de recherche s'inscrivent dans cette dernière optique. Autrement dit, ils portent sur la conception d'algorithmes d'allocation de ressources tout en prenant en compte les spécificités du Cloud pour répondre aux besoins des systèmes de workflows. Plus précisément, nous avons proposé trois approches multi-critères complémentaires pour la gestion des ressources dans un environnement de Cloud, en tenant compte de deux critères de qualité de service : (i) temps d'exécution et (ii) coût d'exécution.

Pour récapituler, l'objectif de cette thèse est de proposer un modèle pour une utilisation optimisée des ressources déployées dans le cadre du *Cloud Computing*. Ainsi la thèse tentera de répondre aux questions suivantes en ce qui concerne la modélisation et la gestion optimisée des ressources pour l'exécution des processus scientifiques et métiers :

1. Comment prendre en compte le modèle économique du *Cloud* ?
2. Comment prendre en compte les spécifiés du *Cloud* telles que « l'illusion de res-

sources infinies » et l'élasticité ?

3. Quelle est la meilleure approche à utiliser pour la résolution des problèmes d'allocation des ressources et d'ordonnancement des tâches des processus ?
4. Les approches proposées dans le cadre des grilles de calcul sont-elles adaptées dans le cadre du *Cloud Computing* ?
5. Comment doit-on gérer le fait que plusieurs instances d'un processus métier peuvent avoir accès de manière concurrente à un ensemble de ressources ?
6. Que devient la notion d'équité lorsque plusieurs critères d'optimisation sont pris en compte ?

La question centrale, sur l'ordonnancement des tâches d'un processus en utilisant des ressources hétérogènes, à laquelle nous nous sommes intéressés n'a pas laissé la communauté des grilles de calcul indifférente. Plusieurs études apportent une réponse satisfaisante quant à l'utilisation optimisée des ces ressources. Cependant, comme mentionné précédemment, les approches proposées ne sont pas adaptées au contexte du *Cloud*. Pour pallier leurs insuffisances, les travaux présentés dans cette thèse visent répondre aux questions, interdépendantes listées ci-dessus.

2.4 Contributions

L'objectif principal de cette thèse est de mettre à la disposition des utilisateurs des ressources, déployées dans l'environnement du *Cloud Computing*, et des méthodes permettant de les aider à choisir le meilleur fournisseur pour exécuter chacune des tâches constituant un processus donné. Dans ce contexte, nous nous sommes intéressés à l'exécution de deux types de processus : (1) les processus scientifiques (ou workflow scientifique) et (2) les processus d'entreprise (processus métier). Lorsque l'on considère le dernier type de processus, plusieurs instances peuvent avoir accès, de manière concurrente, aux ressources disponibles. Deux critères de qualité de service des plus importants ont été pris en compte, à savoir : (1) le temps d'exécution et (2) le coût engendré par l'utilisation d'un ensemble de ressources. En outre, lorsque plusieurs instances d'un même processus s'exécutent en parallèle un troisième critère est requis. Son objectif est d'assurer un accès équitable aux différentes ressources utilisées. La Figure 2.1 résume l'ensemble de nos contributions. Soit un ensemble de tâches d'un processus nécessitant un ensemble de ressources machines virtuelles et/ou humaines pour leur exécution. Notre approche propose des stratégies d'allocation de ressources optimisant à la fois le coût global d'exécution et

le temps d'exécution liés à l'utilisation de ces ressources, tout en prenant en compte le temps et le coût de transfert entre les différentes ressources utilisées.

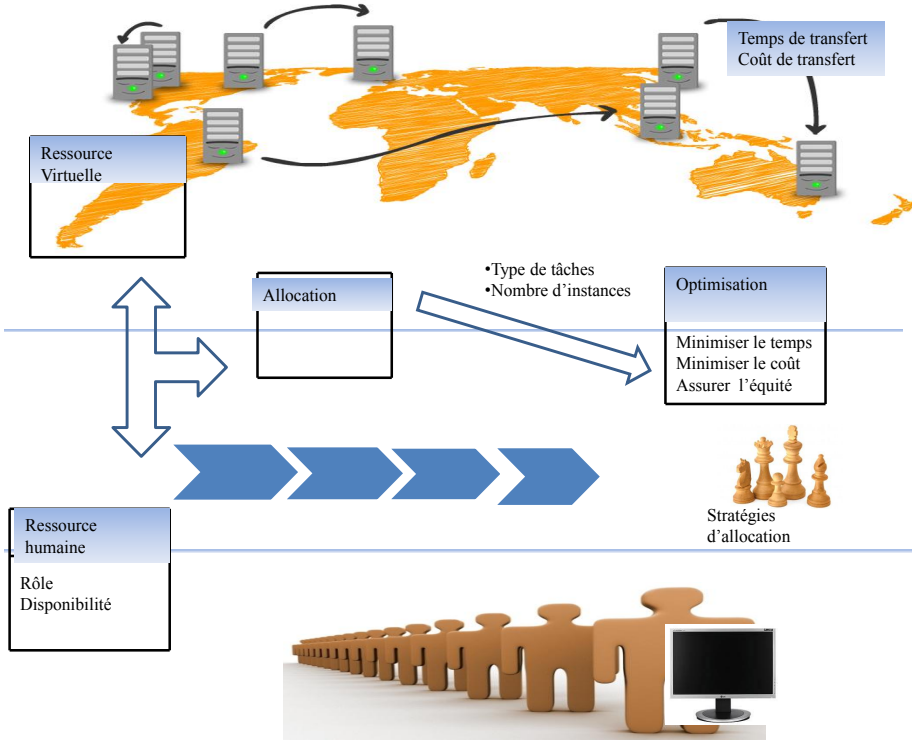

FIGURE 2.1 – Résumé de l'ensemble de nos contributions

Notre point de départ sont les études [77][78][79][80][125] présentées dans le cadre de l'ordonnancement de graphes acycliques de tâches en présence de ressources hétérogènes. Comme mentionné précédemment, les approches proposées dans ces études présentent de nombreuses insuffisances dont les plus significatives sont : (i) la non prise en compte des spécificités du *Cloud Computing* dont le modèle économique et (ii) l'élasticité qu'offre ce nouveau paradigme. Ces études supposent que le nombre de ressources mises à disposition des utilisateurs sont en nombre fini. Elles ne considèrent qu'un seul critère d'optimisation (le temps d'exécution). De plus, elles ne s'intéressent qu'à l'exécution d'une seule instance du processus en question où toutes les tâches sont supposées automatisées. Afin de pallier ces insuffisances, nous avons proposé un modèle bi-objectif, en prenant en compte deux critères conflictuels (le temps et le coût d'exécution) et en considérant le fait qu'un utilisateur peut en principe demander autant de ressources (machines virtuelles) nécessaires pour l'exécution de ses tâches. **Plus précisément, nous avons abordé le problème sous trois aspects complémentaires**.

Notre première contribution consiste à proposer un ensemble de stratégies pour

l'allocation des ressources et l'ordonnancement des processus scientifiques. Nous avons ainsi pris en compte le modèle économique du *Cloud Computing* en introduisant la fonction coût induit par l'utilisation d'un ensemble de ressources ainsi que la caractéristique d'*illusion de ressources infinies*. Autrement dit, le nombre de machines virtuelles est supposé être en nombre infini. Les apports de cette première contribution se déclinent selon les points suivants :

1. La formulation du problème d'allocation de ressources et d'ordonnancement des tâches constituant un processus scientifique dans le cadre du *Cloud Computing*.
2. La proposition d'une première approche, d'allocation de ressources et d'ordonnancement d'un processus scientifique, basée sur la fonction objectif temps d'exécution.
3. La proposition d'une deuxième approche basée sur la fonction objectif coût engendré par l'utilisation d'un ensemble de ressources disponibles.
4. Une troisième approche a été proposée combinant les deux premières approches. Plus précisément, l'objectif de cette dernière est de prendre en compte les deux critères de qualité de service simultanément.
5. Étant donné l'appartenance du problème traité à la classe de problème NP-complet et afin d'évaluer la qualité des solutions obtenues en utilisant notre approche, nous avons proposé deux bornes inférieures des fonctions objectifs temps et coût d'exécution.

Notre deuxième contribution est une extension des travaux présentés dans la première contribution dont l' objectif est de prendre en compte le fait que toutes les tâches constituant un processus ne sont pas toujours automatisées et qu'une intervention de ressources humaines est souvent indispensable pour les accomplir. D'autre part, étant donné l'opacité (une ressource peut être éventuellement utilisée par d'autres instances de processus sur lesquelles on a aucun contrôle) qui régit la gestion des ressources disponibles, nous avons proposé l'utilisation des modèles de prévision afin de prédire les disponibilités des ressources partagées. Les contributions de cette deuxième partie peuvent être résumées comme suit :

1. La proposition d'une première approche, basée sur le temps d'exécution, d'allocation de ressources informatiques et humaines et d'ordonnancement de tâches d'un processus scientifique. Remarquons que les ressources humaines peuvent être sollicitées par d'autres instances d'où l'utilisation d'un modèle de prévision pour la disponibilité de ces ressources.

2. La proposition d'une deuxième approche basée sur le coût induit par l'utilisation d'un ensemble de ressources.
3. La prise en compte des deux critères de qualité de service temps et coût d'exécution simultanément, en se basant sur les deux premières approches.

Notre troisième contribution est une généralisation des deux contributions précédentes. Plus précisément, notre objectif est de proposer des méthodes d'allocation de ressources et d'ordonnancement de plusieurs instances d'un même modèle de processus métier. Ceci est justifié par le fait que souvent plusieurs instances du même processus métier sont lancées en parallèle. Par conséquent, le temps d'exécution d'une instance lorsque toutes les ressources disponibles sont à sa disposition est forcément plus petit que son temps d'exécution lorsque ces ressources sont partagées. Afin d'assurer l'équité dans le partage des ressources, nous avons introduit un troisième critère appelé *équité*. Son objectif est de déterminer la priorité de chacune des instances en cours, de façon à ce que l'augmentation du temps d'exécution soit la même pour toutes les instances. Autrement dit, l'objectif est de ne pas retarder l'exécution d'une instance par rapport aux autres instances.

En résumé, étant donné un ensemble d'instances d'un modèle de processus métier, exécutées de manière concurrente, un ensemble de ressources informatiques (machines virtuelles) et un ensemble de ressources humaines, les différents challenges auxquels nous nous sommes intéressés se déclinent en trois points :

1. Quelle est l'approche d'optimisation la plus appropriée pour l'allocation de ces ressources et l'ordonnancement de ces instances ?
2. Comment peut-on quantifier le critère d'équité tout en prenant en compte deux critères de qualité de service (temps d'exécution et coût d'exécution) ?
3. Peut-on proposer un algorithme efficace étant donné l'appartenance du problème traité à la classe de problèmes NP-complet ?

Pour répondre à ces différents challenges, nos contributions peuvent être résumées ainsi :

1. Nous avons proposé un modèle bi-objectif pour la résolution du problème d'allocation de ressources et d'ordonnancement d'un ensemble d'instances d'un processus métier.
2. Nous avons introduit un troisième critère d'optimisation permettant de mesurer l'équité quant au partage de ressources par plusieurs instances du même processus.

3. Nous avons proposé plusieurs stratégies pour l'allocation des ressources et l'ordonnancement des instances d'un processus métier. De plus, nous avons distingué deux cas de figure, à savoir :

   (a) toutes les instances arrivent au même instant dans le système.

   (b) les instances arrivent au fur et à mesure dans le système.

2.5 Conclusion

Le phénomène de migration des applications d'entreprise sur Internet est plus rapide que jamais avec l'émergence du *Cloud Computing*. Ainsi, le passage des modèles logiciels traditionnels au *Cloud Computing* n'a cessé de s'accentuer ces dernières années à cause de ses nombreux avantages énoncés. Cependant, ce paradigme est confronté à de nombreux problèmes qui peuvent compromettre son succès commercial. Nous avons évoqué plus particulièrement le problème d'ordonnancement des tâches des processus en présence de multiples fournisseurs de services comme contexte général de notre travail. En présence de ces offres multiples, nous avons souligné l'importance de mettre à la disposition des utilisateurs un outil d'aide à la décision les guidant dans le choix du meilleur fournisseur pour exécuter chacune des tâches constituant un processus donné.

Ensuite, nous avons exposé notre problématique de recherche qui consiste en l'optimisation bi-objectifs d'exécution des processus lorsque l'on utilise des ressources déployées dans le cadre du *Cloud Computing*. Nous avons évoqué l'importance de distinguer deux cas de figure : (1) toutes les tâches du processus en question sont automatiques et (2) l'intervention de ressources humaines est indispensable pour exécuter certaines tâches du processus. Enfin, nous avons exposé les différentes contributions que nous avons développées tout au long de cette thèse pour répondre aux problèmes posés.

Dans le chapitre suivant, nous allons introduire les notions de base nécessaires à la compréhension des trois contributions de cette thèse. Nous enchaînerons, dans le chapitre 4 sur l'introduction du problème d'ordonnancement des graphes acycliques de tâches. Nous présenterons aussi les différentes approches candidates pour la résolution de la problématique à laquelle nous nous sommes intéressés et les limitations des approches existantes.

Chapitre 3

Concepts et état de l'art

Sommaire

3.1 Introduction

Au niveau des processus et des fonctions métiers[72][76], les entreprises cherchent avant tout la performance, le partage des ressources, une meilleure coordination inter-processus ainsi qu'une collaboration plus étroite ce qui a convaincu de nombreuses entreprises à adopter le *Cloud* [55][56].

Par ailleurs, certains des plus grands succès du *Cloud* concernent des solutions de collaboration. En effet, l'un des avantages du *Cloud* est la favorisation de la coordination des processus et des fonctions métiers. Si bien que de nombreuses entreprises l'on rapidement adopté pour l'exécution de leurs processus.

Les principaux avantages opérationnels qu'offre le *Cloud* se situent au niveau de la baisse des coûts de production des services informatiques grâce à la disponibilité et l'élasticité des ressources informatiques ainsi qu'à des systèmes de facturation portant sur la consommation réelle des utilisateurs de ses ressources.

La suite est organisée comme suit. La section suivante présente le paradigme du *Cloud Computing* et discute de ses avantages et inconvénients pour l'exécution des applications d'entreprises. L'objectif de cette thèse étant l'allocation de ressources et l'ordonnancement de tâches des processus, nous survolons dans la Section 3.3 les notions et les définitions des éléments de base nécessaire à la compréhension de ce mémoire tels que les workflows et les systèmes de gestion de workflows. La Section 3.4 est consacrée à la présentation du socle du *Cloud* et du rôle de l'Architecture Orientée Service dans l'interopérabilité des systèmes d'information de plus en plus hétérogènes. La Section 3.5 conclut le chapitre.

3.2 Cloud Computing

Le terme du *Cloud Computing* vient du monde des télécommunications. Lorsque, en effet, il fallait dessiner des processus ou des diagrammes, les réseaux de télécommunications et Internet sont généralement représentés par un nuage. Le mot nuage symbolise le traitement des informations sans aucune intervention directe et surtout sans savoir ce qui s'y passe concrètement [162].

Si l'origine de son appellation ne fait pas débat, le reste de ce que peut engendrer le mot *Cloud* fait polémique et plusieurs questions se posent dont :

— est-ce une application utilisable à la demande ?

— est-ce du partage de ressources ?

— et est-ce du management des processus ?

Les réponses à ces questions se trouvent dans de nombreuses définitions du *Cloud Computing*. Nous retenons dans ce rapport les deux définitions proposées par NIST [11] en 2009 :

« Le *Cloud Computing* est l'ensemble des disciplines, technologies et modèles commerciaux utilisés pour délivrer des capacités informatiques (logiciels, plateformes, matériels) comme un service à la demande ».

« Le *Cloud Computing* est un modèle d'accès à des ressources informatiques partagées et configurables (par exemple, réseaux, serveurs, stockage, applications et services), depuis un accès réseau, à la demande, de manière simple, à partir de quel type d'appareil et depuis n'importe quel endroit. Ces ressources sont disponibles rapidement et opérationnelles avec un minimum d'effort et d'interactions avec le fournisseur de service ».

Au delà de cette définition qui introduit les premiers termes propres au *Cloud Computing*, le NIST la complète par les éléments suivants :

1. Cinq caractéristiques essentielles : (i) mise en commun des ressources (ou partage des ressources), (ii) élasticité des ressources, (iii) des services à la demande, (iv) services mesurables et (v) une bande passante élevée.
2. Trois modèles de services : (i) Infrastructure as a Service (IaaS), (ii) Plateforme as a Service (PaaS) et (iii) Software as a Service (SaaS).
3. Quatre modèles de déploiement : (i) *Cloud public*, (ii) *Cloud privé*, (iii) *Cloud hybride* et (iv) *Cloud communautaire*.

Nous donnons un peu plus de détails sur ces éléments dans ce qui suit.

3.2.1 Les caractéristiques du *Cloud*

Les cinq caractéristiques du *Cloud* sont représentées par la Figure 3.1. Elles se définissent de la façon suivante :

— Mise en commun des ressources ou partage des ressources (*pooling*) : les ressources mises à la disposition des utilisateurs en utilisant une plate-forme de type *Cloud* sont accessibles via Internet quelque soit le périphérique utilisé (PC, Mac, tablette, SmartPhone...). Ces ressources peuvent être partagées par plusieurs utilisateurs. Les fournisseurs s'appuient sur le concept de virtualisation, c'es-à-dire les ressources sont allouées et libérées selon les besoins des clients. Ceci a pour conséquence de

11. National Institute of Standards and Technology : http://www.nist.gov

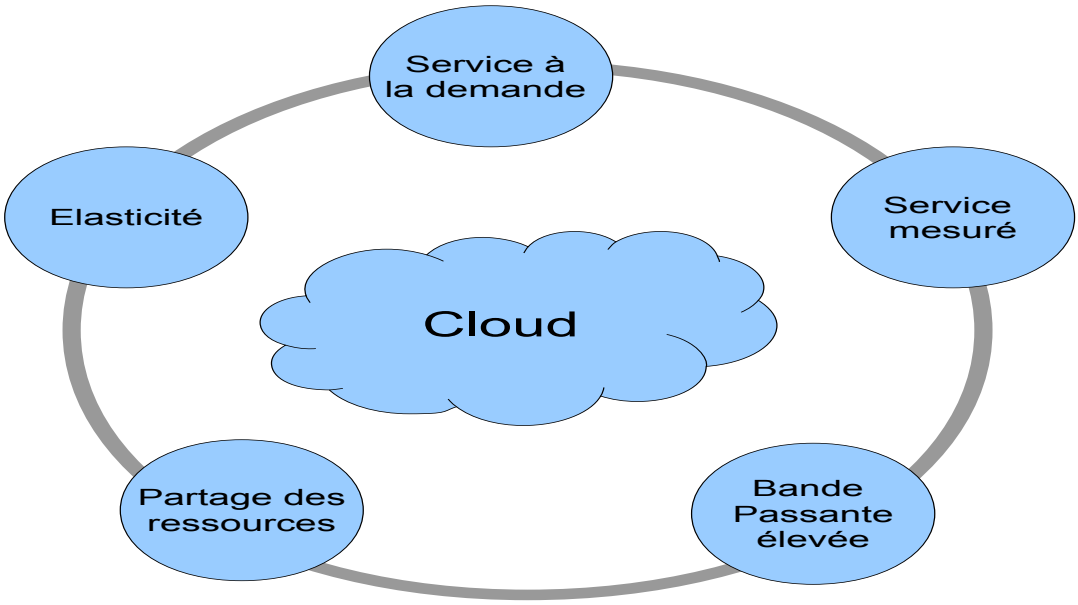

FIGURE 3.1 – Les cinq caractéristiques du *Cloud Computing*

masquer la localisation physique des resources pour les présenter sous forme de machines virtuelles.

— Élasticité : il est possible, grâce au *Cloud*, de disposer de plus de ressources instantanément pour soutenir une forte demande [12]. Inversement, par exemple en cas de diminution du nombre d'utilisateurs d'une plate-forme, il est possible de libérer des ressources. Autrement dit, il est possible avec *Cloud* d'acquérir de nouvelles ressources en fonction de l'évolution des besoins métiers, et le client peut soit augmenter ou diminuer par exemple le nombre de machines et/ou leurs puissances. Généralement, cette tâche est automatisée et c'est l'opérateur de service qui ajuste les ressources en fonction des besoins.

  Du point de vue du client (utilisateur), les ressources du *Cloud* doivent paraître illimitées, accessibles en n'importe quelle quantité et de n'importe quel endroit.

— Service à la demande : le *Cloud* permet d'utiliser des ressources sans pour autant avoir besoin de faire une demande d'intervention des fournisseurs. En pratique, les utilisateurs accèdent via une interface Web à des services informatiques tels que la capacité supplémentaire de stockage et/ou de calcul sans intervention humaine (cette tâche étant automatisée) de la part de chaque fournisseur de services.

— Service mesuré : dans un environnement de type *Cloud*, le fournisseur de service est

12. Par exemple pour répondre à l'augmentation du nombre d'utilisateurs d'une plate-forme Web d'e-commerce.

Modèle	vCPU	Mémoire (Gio)	Stockage sur SSD (Go)
m3.medium	1	3,75	1 × 4
m3.xlarge	4	15	2 x 40
m3.2xlarge	8	30	2 x 80

TABLE 3.1 – Types de machines virtuelles du fournisseur de service Amazon EC2 pour la famille dite M3. Source : http://aws.amazon.com/fr/ec2/instance-types/

capable de mesurer de façon précise la consommation des utilisateurs des différentes ressources mises à sa disposition (par exemple CPU, stockage, bande passage, etc), ce qui lui permet de facturer à l'usage ses clients. Parmi les services facturés, on peut citer : (i) la quantité d'espace de stockage utilisé, (ii) le nombre de transactions, (iii) la bande passante, (iv) le nombre d'entrées et de sorties et (v) la puissance de calcul utilisée.

— Bande passante élevée et accès réseau universel : les ressources mises à disposition des utilisateurs sont accessibles via Internet de n'importe quel endroit et en utilisant des plateformes hétérogènes (PC, téléphone, mobile, PDA, ordinateur portable, etc.) grâce à la capacité actuelle des réseaux.

Nous terminons cette partie par donner un exemple concret des différents types de machines virtuelles (ressources) mises à la disposition des utilisateurs en utilisant par exemple les instances d'*Amazon Elastic Compute Cloud EC2* (aws.amazon.com). Ce fournisseur met à la disposition des ses clients un large éventail de types d'instances de machines virtuelles qui correspondent à une combinaison en matière de capacité de CPU, de mémoire, de stockage et de mise en réseau. Parmi les machines virtuelles les plus utilisées, nous pouvons cité la famille dite *M3* destinée à un usage général et composée de quatre types de machines virtuelles décrites par le tableau 3.1.

3.2.2 Les modèles de services du *Cloud*

Pour un maximum de flexibilité, de rapidité et d'efficacité et afin de tirer le meilleur des investissements informatiques, les entreprises s'appuient de plus en plus sur des services applicatifs. Les fournisseurs de services rendent leurs prestations accessibles aux usagers à partir d'interface Internet. L'objectif du *Cloud* est d'externaliser ces prestations d'infrastructure informatique nécessaire à l'hébergement de ces services. Cette gestion de l'infrastructure est proposée sous la forme de service (*As A Service*), par les fournisseurs

d'infrastructure.

Les services applicatifs décrivent les différentes catégories de services accessibles depuis une plateforme de *Cloud Computing*. Ceux sont les types de services que le fournisseur est capable de proposer. Le *Cloud* offre trois grandes familles de déploiement de services qui est une continuité de l'informatique traditionnelle mais à grande échelle. La Figure 3.2 donne une correspondance entre le modèle informatique traditionnel et le *Cloud Computing*.

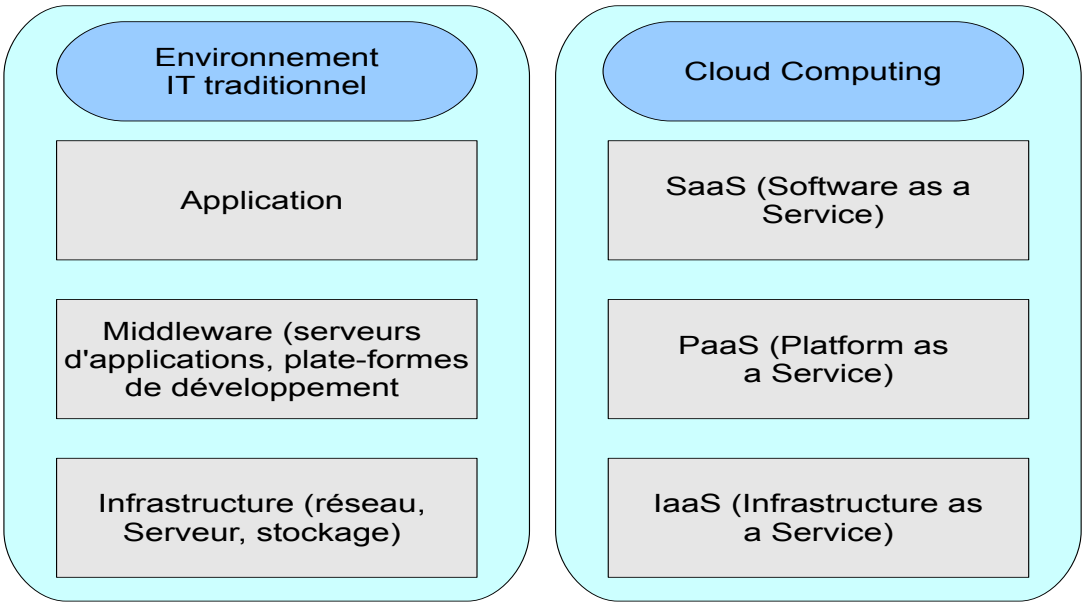

FIGURE 3.2 – Comparaison entre un modèle informatique traditionnel et le *Cloud Cmputing*

1. Infrastructure et IaaS

Généralement trois couches sont à distinguer lors du déploiement d'une application : l'infrastructure, le middleware et l'application elle-même. Dans un environnement traditionnel, la couche d'infrastructure représente la couche matériel (ou le matériel physique). Dans certains cas, cette couche physique peut être virtualisée et fournit une puissance de calcul , le réseau, le stockage et les systèmes d'exploitation.

Dans un environnement du *Cloud*, le modèle IaaS (*Infrastructure as a Service*) représente la couche d'infrastructure composée principalement d'environnement virtualisé mettant à la disposition des utilisateurs une puissance de calcul, de réseau, de stockage, les systèmes d'exploitation et l'hébergement sous la forme d'un service. Dans ce modèle de

service, le fournisseur met donc à la disposition des utilisateurs une plateforme d'hébergement exploitée par un opérateur et accessible via Internet. Autrement dit, un prestataire de service propose la location de composants informatiques tels que des espaces de stockages, une bande passante, des unités centrales et des systèmes d'exploitation. Les utilisateurs d'une IaaS peuvent donc utiliser à la demande des serveurs virtuels situés dans des centres de données (*datacenters*) sans avoir à gérer les machines physiques (remplacement de matériels, climatisation, électricité, ...).

Parmi les fournisseurs des solutions IaaS, nous pouvons citer *Amazon Elastic Compute Cloud (EC2)*, Eucalyptus, Flexiscale, Linode, RackSpace Cloud et Terremark. Avec Amazon, par exemple, considéré comme un fournisseur classique de solutions IaaS, un client peut accéder à un ordinateur sous la forme d'une image d'une machine virtuelle sur laquelle il installe un système d'exploitation et des applications.

2. Middleware et PaaS

Dans les environnements traditionnels, le middleware représente le serveur d'application, le serveur Web et les systèmes de base de données.

Le modèle PaaS du *Cloud* représente la couche de middleware sur laquelle la plateforme de support est déjà installée pour le client. Les choix techniques ont été déjà faits concernant les langages de programmation, les applications et les environnements de base de données. Autrement dit, dans ce modèle, le fournisseur met à disposition des clients des machines virtuelles, des systèmes d'exploitation, des applications, des services, des plateformes. Le client peut alors soit utiliser les applications qui sont programmées à partir des langages et des outils supportés par le fournisseur de service PaaS, soit installer ses propres outils. Le fournisseur gère l'infrastructure PaaS, le système d'exploitation et les logiciels associés. Le client est responsable des applications qu'il déploie.

Parmi les fournisseurs des solutions PaaS, nous pouvons citer Force.com, GoGrid CloudCenter, GoogleAppEngine ou encore Windows Azur Platform.

3. Applications et SaaS

Le modèle SaaS représente l'accès à une application informatique et aux services associés. La différence entre SaaS et logiciel est essentielle. En effet, les SaaS proposent des logiciels opérationnels, prêt à l'emploi, sans passer par une étape d'installation et sans aucune tâche de maintenance. Concrètement, lorsque un client sollicite une solution SaaS, il se connecte simplement au service depuis un navigateur.

Parmi les revendeurs de solutions SaaS figurent GoogleApps, Oracle on Demand, Salesfirce.com et SQL Azure.

Quelque soit le modèle utilisé, les fournisseurs proposent des contrats de service (SLA [13]) qui couvrent non seulement les aspects de performance du matériel et du logiciel, mais qui s'étendent également à la protection des données qu'ils stockent pour leurs clients.

Par ailleurs, il existe dans la littérature une autre classification des services offerts dans le cadre du *Cloud Computing*. Cette classification est donnée par la Figure 3.3 :

1. IaaS destinée aux experts techniques et représente la couche réseau du *Cloud* ;
2. PaaS destinée aux développeurs ;
3. et SaaS destinées aux utilisateurs (ou clients).

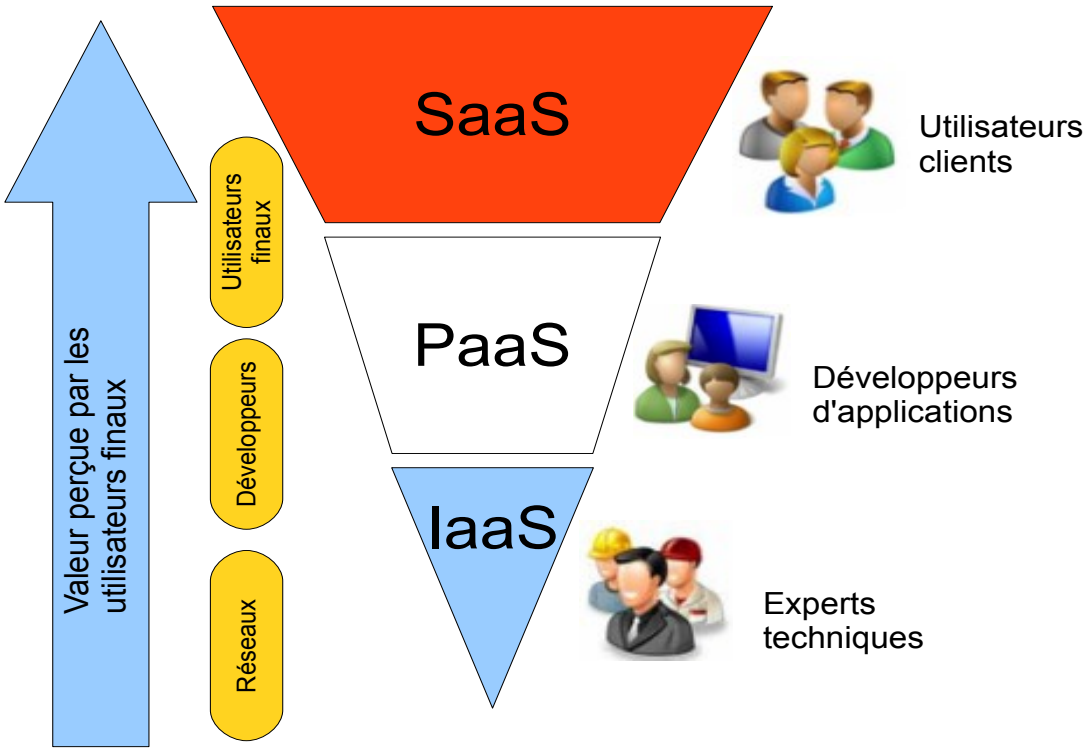

FIGURE 3.3 – Valeur perçue par les utilisateurs des trois modèles de déploiement du *Cloud*

13. Service Level Agreement est un document la qualité de service requise entre un client et son fournisseur de service

3.2.3 Les modèles de déploiement du *Cloud*

On distingue quatre modèles de déploiement du *Cloud*. Chacun des ces modèles correspond à une architecture particulière décrite ci-après. Ces modèles de déploiement sont représentés par la Figure 3.4.

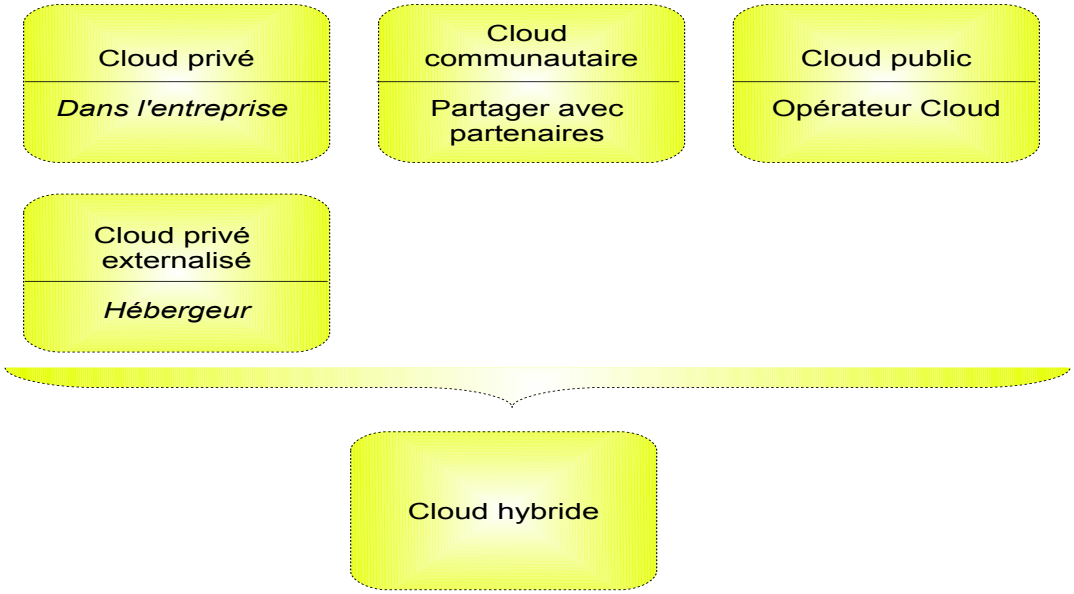

FIGURE 3.4 – Les différents modèles de déploiement d'un *Cloud*

1. Le Cloud public

Les clients utilisant ce modèle de déploiement ont accès à des services du *Cloud* via Internet public sans savoir précisément où sont hébergées leurs données ni où sont exécutés leurs traitements. Dans la pratique, les données des utilisateurs peuvent passées d'un centre de données à un autre afin d'optimiser les capacités du prestataire, ce qui peut poser de sérieux problèmes de sécurité. Néanmoins, l'un des avantages de ce type de déploiement est son coût raisonnable de mise en œuvre. En effet, les coûts de déploiement du matériel, des applications et du réseau sont pris en charge par le fournisseur qui les répartit sur plusieurs clients. Ce type de déploiement offre aussi la possibilité de faire évoluer les ressources et le métier.

Par ailleurs, il est important de noter que ce type de déploiement est celui qui incarne le plus l'élasticité idéale, voire illimitée. Ainsi, un client donné peut avoir autant de ressources que nécessaires pour ses applications.

Parmi les services publics du *Cloud* figurent par exemple les sauvegardes (*backup*[14]), les services de stockage de fichiers ou encore des plate-formes d'infrastructures plus sophistiquées pour le déploiement d'applications, ou de services Web comme les messageries Webmail.

Les principaux prestataires dans le monde du *Cloud* public sont donnés par la Figure 3.5.

FIGURE 3.5 – Quelques exemples des principaux prestataires de services du *Cloud*

2. Le Cloud privé

Si la plupart des entreprises optent pour le *Cloud public*, certaines tirent profit de l'application des principes du *Cloud Computing* en interne. Ceci, en organisant des centres de données en s'appuyant sur les techniques de virtualisation permettent notamment de gagner en élasticité et une optimisation dynamique des ressources utilisées. Ainsi, une entreprise peut s'appuyer sur ce type de déploiement afin de faire face à des pics de consommation des ressources mises à disposition des utilisateurs. Ce mode de déploiement s'appelle le *Cloud privé*. En pratique, un *Cloud privé* peut être géré par l'entreprise utilisatrice elle-même ou par un prestataire externe qui met à disposition des utilisateurs un parc de machines tout en s'adaptant à la demande de l'utilisateur. Par ailleurs, une

14. Le *backup* est utilisé pour mettre en sécurité les données contenues dans un système informatique en les dupliqaunt

même infrastructure peut accueillir plusieurs *Cloud privés* virtualisés où chaque client y accède via son propre réseau.

4. Le Cloud communautaire

Ce type de *Cloud* est conçu pour le partage des ressources ou des informations au sein de groupes spécialisés. L'objectif de ce type de déploiement est l'extension du nombre d'applications et de composants d'infrastructures fournis sous la forme de service, ce qui permet de consolider et d'optimiser l'efficience des opérations mises à la disposition des utilisateurs. La plateforme partagées entre plusieurs partenaires peut être gérée par les organisations membres ou un prestataire externe.

Par ailleurs, ce type de déploiement peut avoir les mêmes caractéristiques q'un *Cloud privé* en terme de sécurité et de ressources.

3. Le Cloud hybride

Ce type de déploiement est le résultat d'un constat pragmatique de l'utilisation efficace des ressources disponibles. En effet, aujourd'hui les entreprises ne peuvent pas se passer d'un *Cloud public* à cause de ses nombreux avantages avérés, mais pour autant, elles ne souhaitent pas, notamment pour des raisons de sécurité, externaliser la gestion de leurs données sensibles à des tiers. Comme son nom l'indique, le *Cloud hybride* est une composition de deux ou plusieurs types de déploiement (privé, communautaire, public).

Une comparaison des responsabilités (entreprise et *Cloud*, c'est-à-dire entre le client et son fournisseur de services) est donnée par la Figure 3.6. Cette comparaison est basée sur les éléments suivant :

a. Gestion des applications .
b. Gestion du moteur d'exécution (ou *runtimes*) .
c. Intégration des applications de l'entreprise en s'appuyant sur une architecture orientée services (SOA, *Service Oriented Architecture*).
d. Gestion des bases de données.
e. Utilisation des techniques de virtualisation.
f. Gestion des serveurs.
g. Gestion des supports de stockage.
h. Gestion de l'infrastructure réseau.

A titre d'exemple, lorsque l'on utilise un modèle classique tous ces éléments sont de la responsabilité de l'entreprise. Par contre, lorsque l'on utilise un modèle de déploiement IaaS, uniquement les cinq premiers éléments sont de la responsabilité de l'entreprise et les autres éléments relèvent de la responsabilité du fournisseur de services.

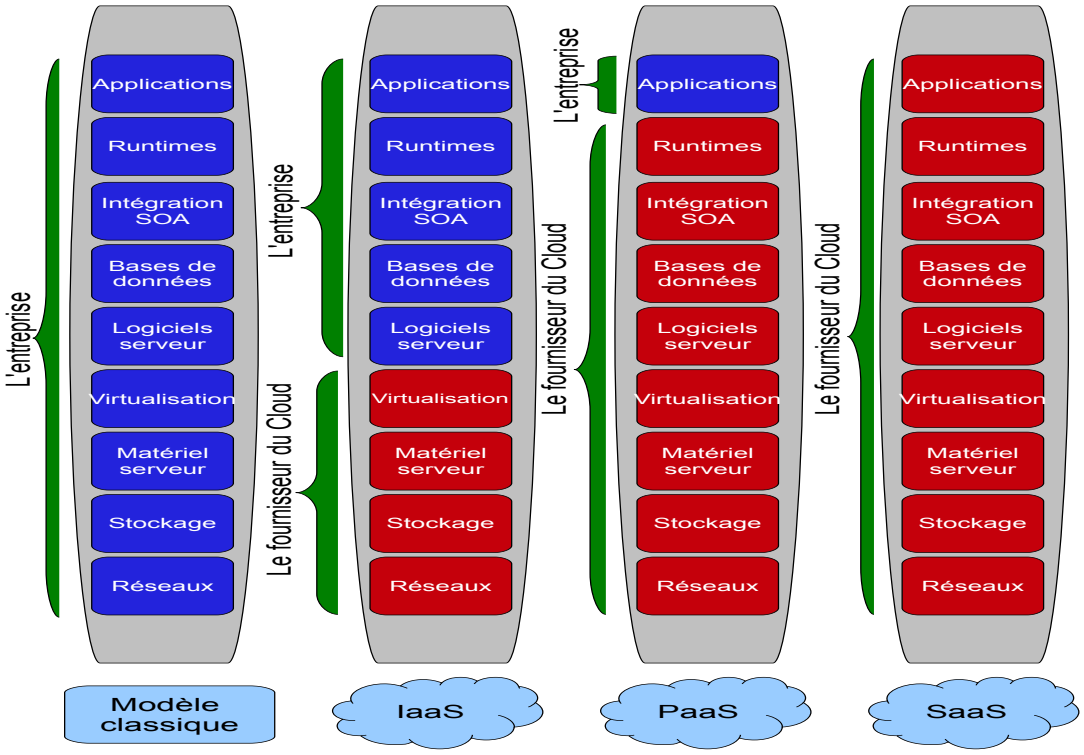

FIGURE 3.6 – Comparaison des responsabilités entre les deux modèles classique et *Cloud*

3.2.4 La dimension économique du *Cloud*

L'emergence du *Cloud* est certe une évolution majeure sur la plan technique. Mais son succès est dû au fait que cette évolution est accompagnée de mouvements significatifs sur la plan économique. En effet, ce qui change avec l'introduction de ce nouveau paradigme est la façon dont les entreprises utilisent (consomment) les ressources informatiques en souscrivant des abonnements leur permettant d'utiliser des ressources, de stockages, de calcul et/ou des applications et des plate-formes de développement, à la demande. Ainsi, les utilisateurs ne seront facturés que pour ce qu'ils consomment réellement et seront assurés d'une mise à disposition élastique des ressources en fonction de leur besoins.

Le modèle économique du *Cloud* est basé sur le *pay-per-use* (paiement à l'utilisation) qui se décompose en deux principales modalités :

1. *Pay-as-you-go* : c'est un modèle d'abonnement basé sur l'utilisation réelle de ressources approvisionnées dynamiquement.
2. *Pay-as-you-grow* : c'est un modèle d'abonnement similaire au précédent mais légèrement moins flexible dans la mesure où la demande ne peut que s'accroître. Néanmoins, il applique des tarifs à la baisse en fonction du franchissement des paliers d'utilisation des ressources.

3.2.5 Intérêts du *Cloud*

Exécuter une application dans un environnement de *Cloud* présente généralement des avantages financiers car son utilisateur n'a aucun investissement initial à prévoir pour la plupart des services utilisés et pas de personnel informatique à gérer. Bien que cette raison soit suffisante pour adopter le *Cloud*, elle n'est pas la seule. En effet, le *Cloud* permet également d'augmenter instantanément l'échelle d'une application en respectant des design patterns appropriés lors de sa conception.

L'un des bénéfices du *Cloud* perçu par les entreprises est la réduction des coûts. En effet, en adoptant le *Cloud* de nombreuses entreprises ont pu réduire **les coûts liés aux équipes d'exploitation**. Cette réduction est possible grâce à la mutualisation et généralement une équipe unique gère les applications de plusieurs entreprises. Un autre avantage de cette mutualisation est l'assurance d'une gestion optimale des ressources. Effectivement, les coût salariaux sont moindres que ceux engendrés par l'utilisation de plusieurs équipes dans lesquelles les ressources sont potentiellement sous-exploitées.

Pour assurer un haut niveau de sécurité et de disponibilité des applications, les fournisseurs disposent de plusieurs centres de données réparties géographiquement. Ce qui facilite encore plus le lissage des coûts induits par cette répartition des données. De plus, les fournisseurs du *Cloud* peuvent négocier les prix à la baisse des serveurs étant le nombre nécessaire pour faire fonctionner un centre de données. Cette négociation est aussi valable pour le coût de l'énergie.

Par ailleurs, de plus en plus d'entreprises sont sensibles au principe de « l'informatique verte » (*green IT*) ce qui s'inscrit dans une démarche de développement durable.

Il existe de nombreux autres avantages à utiliser une infrastructure du type *Cloud* dont une liste non exhaustive décrite ci-après :

1. La standardisation : le *Cloud* offre des services et des prestations automatisées, en libre service et sur une grande échelle, ce qui contribue substantiellement à la réduction des coûts.

2. L'indépendance des clients par rapport aux fournisseurs : le client n'est pas renfermé dans un contrat avec un seul fournisseur et ne paie que pour sa consommation réelle. Il s'agit en effet d'une extension du modèle traditionnel d'*outsourcing* informatique. L'adoption du *Cloud* permet par conséquent une plus grande flexibilité des contrats entre les clients et leurs fournisseurs. Toutefois, il convient de souligner que l'absence de standardisation au niveau des contrats empêche toute comparaison sérieuse entre les prestations de services que l'on trouve sur le marché.
3. La mise en œuvre rapide des services : les solutions offertes dans la cadre *Cloud* sont déjà prêtes à l'emploi. Ainsi, les utilisateurs y accèdent immédiatement dès leur souscription à un ou plusieurs de ces services.

3.2.6 Risques du Cloud

Si le *Cloud* présente de nombreuses opportunités pour les entreprises, de nombreux problèmes peuvent freiner son expansion et compromettre son succès. En effet, plusieurs risques [15] liés à l'utilisation du *Cloud* ont été déjà identifiés et des référentiels comme la norme ISO 27000 aident à les prévoir et à les gérer. La nature de ces risques dépend de nombreux facteurs et peut être liée au modèle de déploiement utilisé (SaaS, PaaS, IaaS). Les grandes catégories de risques liées à l'adoption du *Cloud* se déclinent selon les points suivants :

1. Confidentialité, intégrité et disponibilité (CIA, *Confidetiality, Integrity and Availability*) : les préoccupations majeures lors de la migration des applications vers le *Cloud* sont la confidentialité, l'intégrité et la disponibilité des données. En effet, il est primordial pour une entreprise de prévoir et de prévenir les risques liées à la perte et la confidentialité de ses données. Des clients sont de plus en plus exigeants en ce qui concerne les normes de sécurité et imposent à leurs fournisseurs des certificats comme la norme ISO 27000 ou SAS 70.

   Un autre point de défiance vis-à-vis du *Cloud* est celui de la garantie de l'intégrité des données. Les données doivent être fiables et converties au bon format, sinon de mauvaises interprétations sont susceptibles de se produire.

   Quant la disponibilité, les données déployées dans le *Cloud* doivent être en permanence accessible même lorsque des opérations de maintenances sont effectuées par

15. un risque est défini comme étant un événement qui peut se produire (probabilité non nulle) et avoir un impact positif (opportunité) ou négatif (menace)

les fournisseurs (l'une des solutions utilisées est la réplication des données).

Par ailleurs, il est indispensable de garantir aux utilisateurs la disponibilité des ressources en cas de problèmes (défaillance par exemple) autres que le cas de non disponibilité.

2. Localisation des données : la dématérialisation des données sur différentes sites physique de stockage peut conduire à un éclatement des données et leur répartition sur différentes zones géographiques (par exemple pays).
3. Malveillance : les architectures du *Cloud* sont gérées par des personnes ayant des privilèges très élevés qui sont donc à risque très élevé.
4. Usurpation : deux types d'usurpation peuvent être distingués : (i) usurpation de service offert par l'architecture et (ii) usurpation d'identité d'utilisateurs (clients) de services.
5. Perte de maîtrise et/ou de gouvernance : l'utilisation d'une plate-forme de type *Cloud* peut entrainer la perte de la maîtrise directe du système d'information et une explication opaque de l'infrastructure utilisée. En effet, l'utilisation du *Cloud* est en quelque sorte un renoncement au contrôle de l'infrastructure de calcul et/ou de stockage utilisée.
6. Continuité du métier : autant de questions se posent sur la continuité du métier de l'entreprise qui adopte le *Cloud* telles que : (i) que se passent-ils si le fournisseur de service cessait d'assurer ses services ? et (ii) qu'adviendrait-il de ses données ?

3.3 Les workflows

Un procédé métier a pour objectif de décrire de manière informelle les méthodes de travail d'un groupe de personnes. Concrètement, il est un ensemble d'activités (tâches) ordonnées et exécutées en respectant un ensemble de règles procédurales. Ces tâches sont exécutées dans le cadre d'un environnement organisationnel et technique dans le but de réaliser un objectif précis [40]. Les tâches sont décrites en utilisant un graphe sans cycle dans lequel les nœuds représentent les étapes d'exécution et les arcs représentent le flot de contrôle et de données entre les étapes [41][42][72].

Chaque procédé métier est associée à une seule organisation mais peut interagir avec d'autres procédés métiers appartenant à d'autres organisations.

Tandis qu'un procédé métier permet de décrire de manière informelle les méthodes de travail d'un groupe de personnes, un workflow permet de formaliser, de structurer et d'au-

tomatiser et d'exécuter, dans la mesure du possible, ces méthodes de travail [92][94][96]. La Figure 3.7 donne une vison globale, sous forme d'un méat modèle, de ce qui est un workflow et les concepts associés.

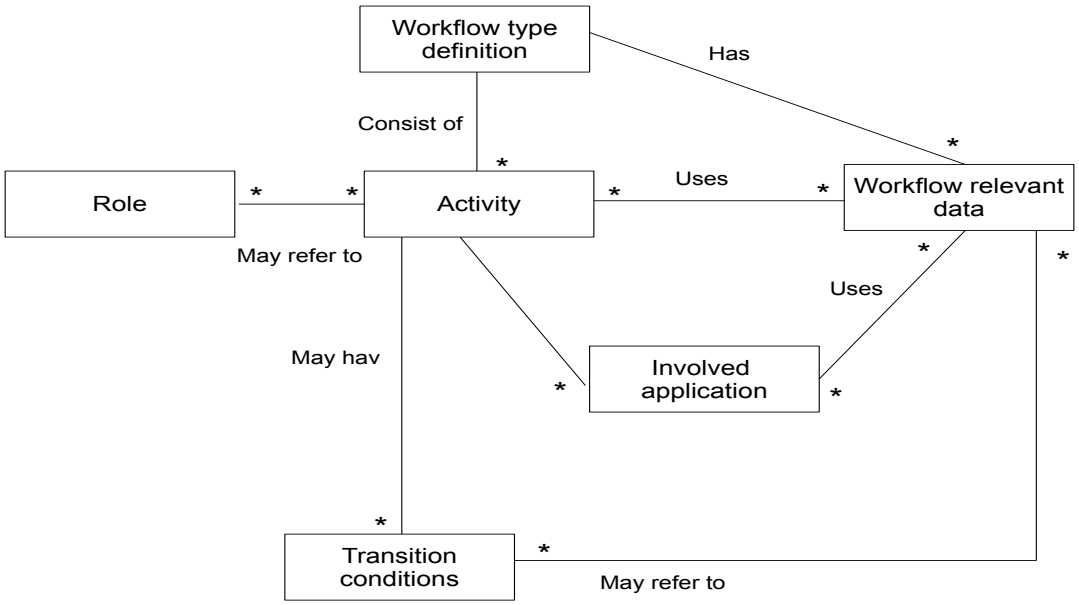

FIGURE 3.7 – Méta modèle de processus. Source : WfMC

Un modèle workflow peut être représenté par un graphe où les nœuds représentent les étapes d'exécution et les arcs représentent le flot de contrôles et le flot de données entre ces étapes. Dans le méta-modèle proposé par la WfMC [16] nous distinguons les composants suivants :

— *Schéma du workflow* : un schéma du workflow est une représentation d'un procédé dans le but d'automatiser sa manipulation (i.e. modélisation et exécution) par un système de gestion de workflow. Concrètement, il est constitué d'un ensemble (réseau) d'activités avec une représentation des dépendances entre elles et des critères spécifiant le démarrage et la terminaison d'un procédé ainsi que des informations sur les activités telles que les compétences requises, les données utilisées, etc. Les données d'entrée et de sortie des activités et des procédés sont représentées sous forme d'ensembles d'éléments de données appelées conteneurs.

— *Activité* : une activité constitue une étape dans un workflow. Chaque activité est caractérisée par : (i) un nom, (ii) un type, (iii) des pré conditions, (iv) des post

16. La Workflow Management Coalition est un groupe de compagnies qui travaillent ensemble pour définir des normes et standards permettant essentiellement l'interoperabilité et l'interconnexion entre les différents produits de gestion de workflows

conditions, (v) des contraintes d'ordonnancement et (vi) un conteneur des données en entrée (données requises pour débuter son exécution) et en sortie (données produites après son exécution).

— *Données* : un ensemble de données sont associées à chaque workflow telles que : (i) les données en entrée des activités, (ii) les données en sortie des activités, (iii) les données requises pour l'exécution des tâches, (iv) les données produites après exécution des tâches, (v) les données nécessaire pour l'évaluation des conditions et (vi) les données échangées entre les activités.

— *Flot de données* : les informations échangées entre les différentes activités constituant un workflow sont représentées par un flot de données. Ce dernier est spécifié via des conteneurs de données entre les activités ayant des conteneurs de données en sortie et des conteneurs de données en entrée.

— *Flot de contrôle* : un modèle de procédé décrit chaque unité de travail à exécuter et la séquence adéquate pour atteindre un objectif donné. Un flot de contrôle permet de décrire l'ordre dans lequel les tâches sont exécutées en plus de la spécification des connecteurs de contrôle entre les activités en utilisant des patrons de contrôle tels que : *le patron séquence*, *le patron du choix exclusif*, *le patron du choix multiple*, *le patron d'exécution parallèle*, etc.

3.3.1 Différence entre un workflow métier et un workflow scientifique

En raison de la différence entre les objectifs visés par les systèmes de gestion de workflows métiers et les systèmes de gestion de workflows scientifiques, ils (les deux systèmes) n'offrent pas les mêmes fonctionnalités à leurs utilisateurs respectifs. Plusieurs travaux ont été proposés afin d'utiliser la technologie des workflows d'entreprise dans le domaine des workflows scientifiques [43][44][45][46]. Plus précisément, la plupart de ces travaux visent à adapter les outils de modélisation et les moteurs d'exécution de workflows d'entreprise au cas des workflows scientifiques [95][109]. Mais il est de toute évidence un écart entre ces deux types de workflows [47]. Les systèmes de gestion de workflows scientifiques assistent les scientifiques dans l'exécution de leurs applications. Le principal intérêt est dans l'automatisation et la parallélisation des tâches constituant le workflow en question. Les systèmes de gestion de workflows métiers se focalisent sur plusieurs aspects simultanément [98][107][108][99]. En effet, ils permettent de définir des flux de contrôles complexes, en utilisant notamment les différents patrons de contrôles et de données, et offrir la pos-

sibilité de gérer d'éventuels dysfonctionnement susceptibles de se produire au moment de l'exécution du workflow en question. De plus, ils offrent la possibilité de créer des tâches qui ne peuvent être exécutées que par des ressources humaines malgré que la gestion et l'exécution du workflow lui-même sont du ressort des systèmes de gestion de workflows. La figure 3.8 suivante représente les points communs et les points de divergences entre les workflows scientifiques et les workflows métiers [144].

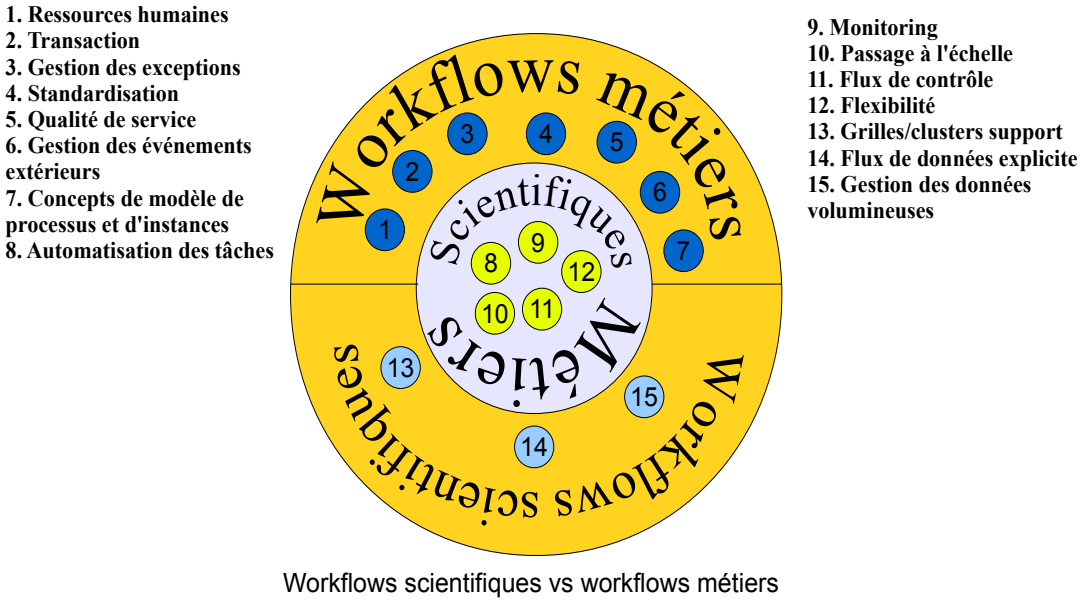

FIGURE 3.8 – Différences entre un workflow scientifique et un workflow métier

Nous comparons dans ce qui suit les deux types de workflows selon cinq aspects, à savoir : (i) les données utilisées, (ii) les traitements effectués, (iii) les acteurs qui interviennent dans la phase d'exécution, (iv) les modèles utilisés pour les représenter et (v) les modes d'exécution utilisés.

Les données

Les données utilisées dans les workflows scientifiques sont généralement complexes que ce soit du point de vu de leur organisation ou du point de vu de leur structure. Ainsi, chaque organisme utilise ses propres spécifications pour les représenter et les organiser. Par conséquent, des parseurs spécifiques doivent être implémentés afin de lire et de manipuler ces données. Dans le but d'assurer l'intéropérabilité entre les données des différents organismes impliqués dans l'exécution du workflow en question des adaptateurs (convertisseurs) doivent être utilisés (c'est le cas, par exemple, lorsque l'on utilise ETL, des

Critères de comparaison	Workflow métier	Workflow scientifique
Champs d'application	Intra-entreprise, organisation importante	Spécifique, ou une équipe de spécialistes
Données manipulées	Faible volume et structure simple	Volumineuses, structure complexe et onéreuses
Nature des traitements	Propriétaire	Nombreux, paramétrables
Intervention des ressources humaines	Intervention importante	Peu d'intervention les tâches sont souvent automatisées
Nature de l'exécution	Souvent de type Chorégraphie	souvent de type orchestration
Modèles	Complexes	Relativement simple
Environnement	Standards pour les utilisateurs	Plateforme complète avec gestion des ressources
Nombres d'instances	Plusieurs instances en parallèle instances différentes	Une seule instance si plusieurs instances , généralement sont toutes les mêmes

TABLE 3.2 – Les deux catégories d'usage de workflows métiers et scientifiques

wrapper-mediators de données ou de schémas de données [48]). Par ailleurs, ces données sont volumineuses et onéreuses. En effet, à titre d'exemple dans le domaine de la télédétection une image haute résolution peut facilement représenter quelques centaines de gigabytes [49]. Dans le cas des workflows métiers, les données échangées entre les différents partenaires correspondent souvent à de petits volumes. En effet, la plupart du temps il s'agit d'échange de messages ou de notifications électroniques. De plus, elles sont simples du point de vu structure et rarement soumises à des spécifications. Cependant, il convient de noter que dans de nombreux cas avant qu'un échange ne s'effectue entre partenaires d'importantes manipulations de données peuvent avoir lieu auparavant.

Les acteurs

Pour un worflow scientifique l'importance n'est pas de modéliser les différentes collaborations et les échanges entre les partenaires, mais plutôt que d'illustrer comment les traitements numériques, constituant le workflow en question, sont enchaînés afin de réa-

liser un calcul scientifique. Généralement, la définition de ce type de workflows nécessite l'intervention de plusieurs experts de différentes disciplines. Une fois le workflow est défini et instancier presque aucune intervention humaine n'est nécessaire [50]. Dans le cas de workflows métiers l'intervention et l'interaction entre les ressources humaines, impliquées dans leur exécution, sont très fréquentes. Dans même la définition du workflow, les différents rôles impliqués dans le déroulement des différentes étapes sont identifiés. Un exemple qui illustre bien cette attribution de rôle est celui d'un achat en ligne qui inclue trois acteurs ayant trois rôles différents, à savoir : (1) le client qui passe la commande, (2) la banque qui reçoit un ordre de virement et (3) l'entreprise qui reçoit un ordre de livraison. Afin de différencier de manière claire les fonctionnalités et les responsabilités de chacun des acteurs impliqués dans un processus donné une bonne définition des rôles est indispensable. Elle est même une des clés pour bien gérer les ressources humaines.

Les modèles de représentation

Un workflow scientifique a généralement pour objectif de vérifier des hypothèses en réalisant des expérimentations. Les terminologies utilisées et les objectifs visés par un workflow scientifique sont très peu compréhensibles par des gens qui ne sont pas dans le domaine. Par conséquent, les formalismes utilisés sont spécifiques à chaque domaine. Autrement dit, les formalismes utilisés pour la modélisation ne sont pas standardisés. tandis qu'un workflow scientifique se focalise sur l'illustration des solutions en abordant des problèmes spécifiques, un workflow métier se focalise davantage sur l'aspect d'intégration intra-entreprise et/ou inter-entreprises [51][52]. Ainsi, un workflow métier est généralement le résultat d'une collaboration entre plusieurs partenaires qui le conçoivent et l'approuvent. Ainsi, la représentation graphique d'un workflow métier met l'accent sur la collaboration entre les différents partenaires impliqués et les détails techniques restent le plus souvent implicites.

3.4 Virtualisation

La construction d'une architecture *Cloud* doit répondre à une exigence d'élasticité, ce qui suppose une allocation dynamique des ressources et théoriquement illimitées. De plus, cette architecture doit pouvoir gérer une montée en charge illimitée dans un environnement multi-serveurs avec un nombre de serveurs extrêmement variable et qui doit pouvoir augmenter ou diminuer de manière instantanée [62]. Pour cela, plusieurs études ont été

proposées dans la littérature pour déterminer le nombre de serveurs à utiliser de façon à satisfaire des critères de qualités de service inscrits dans le SLA [65][64][63].

Par ailleurs, l'existence de plusieurs fournisseurs rend l'interpérabilité encore plus difficile.

La construction d'une architecture *Cloud* est donc une problématique éminement technique et se base essentiellement sur deux notions, à savoir la virtualisation et l'Architecture Orientée Service, décrites ci-après.

La virtualisation est un concept qui s'est imposé depuis la deuxième moitié de la décennie 2000-2010 et qui a rendu crédible, au même titre qu'Internet, la notion même du *Cloud*. En effet, la virtulaisation est totalement indissociable de la notion d'élasticité et de montée en charge [17]. La virtualisation est un concept qui permet de subdiviser un serveur en plusieurs machines virtuelles qui opèrent indépendamment et en se partageant les ressources disponibles sur le serveur. L'utilisation des ressources disponibles sur le serveur sont optimisées via un composant appelé *hyperviseur*. *L'hyperviseur* gère l'allocation de ressources pour chaque machine virtuelle (CPU, bande passante et stockage), instanciée sur le serveur, en arbitrant dynamiquement entre les demandes de chaque machine virtuelle, et ce pour chacun des *cores* d'un même processeur. Plus précisément, plus le processeur a de *core*, et plus *l'hyperviseur* a de latitude dans la gestion dynamique des machines virtuelles. C'est précisément ce principe qui assure la montée en charge dynamique et automatique. Ainsi, si une application subit un pic de charge *l'hyperviseur* le détecte et lui affecte les ressources nécessaires de manière à faire face à cette demande.

3.4.1 Le rôle des architectures orientées services dans le *Cloud*

L'Architecture Orientée Services (SOA) [53] est un concept introduit vers la fin des années 90, avec la description de Jini [18] qui est un environnement de découverte dynamique et d'utilisation des services sur un réseau. Cette architecture est vite devenue un référentielle majeur en terme d'intégration et d'architecture dans le domaine de l'informatique hétérogène. Les premières tentatives, telles que CORBA et RMI [54], proposées pour assurer l'interopérabilité des systèmes d'information hétérogènes ont été vite délaissées à cause de leurs nombreux inconvénients dont l'effort considérable que doit fournir

17. Babcock écrit « C'est la virtualisation qui nous fait apparaître les serveurs plus grands qu'ils ne le sont vraiment »

18. La technologie Jini est une architecture utilisée pour la construction des systèmes à partir d'objets et de réseau

les développeurs pour assurer cette interopérabilité. Le modèle SOA a été popularisé avec des standards comme les services Web dans l'e-commerce. Deux types de services peuvent être distingués, à savoir :

— Les services Web étendus : leurs succès s'est construit autour de trois standard, à savoir : (i) SOAP, pour la communication, (ii) WSDL, pour la description et (iii) UDDI pour la recherche. Les messages échangés entre le client et son fournisseur sont basé sur XML.

— Les service Web Restfull est un style architectural permettant d'identifier chaque ressource d'un manière unique. Le format d'échange des messages entre le client et le fournisseur n'est pas basé forcément sur XML.

Les Architectures Orientées Service (SOA) ont permis de tourner les applications vers le métier de l'entreprise et d'accroître la réutilisation des composants. Cependant, certains projet SOA peuvent être complexes mais il n'en demeure pas moins que ces architectures apportent aux systèmes d'information une plus grande agilité. Par conséquent, une entreprise qui doit par exemple développer une nouvelle application peut s'appuyer sur l'existant ce qui lui permet de se comporter de manière agile pour répondre rapidement aux changements métiers.

En outre, les entreprises doivent capitaliser sur leurs investissements applicatifs et d'infrastructures existantes, tout en privilégiant une architecture permettant une évolution du métier.

Parmi les nombreux avantages de l'utilisation d'une Architecture Orientée Service nous pouvons citer :

1. le couplage faible ;
2. une meilleure tolérance aux pannes ;
3. de plus grandes possibilités d'évolution ;
4. une facilité de maintenance ;
5. la possibilité de réutiliser les composants ;
6. la dynamicité ;
7. l'interopérabilité ;
8. et la composition de services web.

L'architecture orientée service a joué un rôle important dans l'émergence du *Cloud*. En effet, l'infrastructure informatique des entreprises se compose d'un ensemble hétérogène de systèmes d'exploitation, d'applications, de plateformes, etc. C'est l'utilisation de ce

type d'architecture qui permet aux entreprises de répondre rapidement aux changements du métier.

3.5 Conclusion

Nous avons présenté dans ce chapitre le paradigme du *Cloud*. Un paradigme relativement récent qui s'appuie sur un modèle économique basé sur la consommation réelle des utilisateurs. Un paradigme vite adopté par de nombreuses entreprises pour exécuter leurs applications.

Cependant, le *Cloud Computing* engendre des problèmes de différentes nature tels que les problèmes de sécurité, les risques liés à la confidentialité et la disponibilité des données ainsi que le problème du choix entre plusieurs fournisseurs afin de satisfaire au mieux les utilisateurs. C'est ce dernier point qui constitue la principale motivation des travaux présentés dans cette thèse. Plus précisément, nous nous intéressons au problème d'allocation de ressources et d'ordonnancement de tâches des workflow en utilisant des ressources déployées dans le *Cloud*. Nous commençons par présenter dans le chapitre suivant un état de l'art sur le problème d'allocation de ressource et d'ordonnancement de tâches des processus en utilisant des ressources réparties et puis nous présentons nos contributions.

Chapitre 4

Allocation de ressources et ordonnancement de tâches

Sommaire

4.1 Introduction

L'un des avantages du *Cloud computing* réside dans la possibilité offerte aux utilisateurs de réaliser des calculs parallèles. Ces derniers consistent à diviser une application en tâches élémentaires réparties sur plusieurs ressources pouvant s'opérer simultanément. L'objectif de cette répartition est l'amélioration des performances des applications ainsi exécutées comparées à une exécution séquentielle.

Par ailleurs, les tâches constituant les applications réparties peuvent être liées par des contraintes telles que les contraintes temporelles et/ou de données. Par conséquent, la gestion optimale des ressources disponibles et l'ordonnancement des tâches sont des aspects fondamentaux dans la parallélisation des applications. L'allocation de ressources et l'ordonnancement des tâches d'une application consistent à déterminer les ressources à affecter à chacune de ces tâches et l'ordre dans lequel elles doivent être exécutées.

Les problèmes d'allocation de ressources et d'ordonnancement de tâches peuvent être classés en deux catégories principales, à savoir [136][137][138][139][140][141][142] :

1. Les problèmes d'ordonnancement hors ligne (*off-line*) pour lesquels les dates d'arrivées des tâches sont connues avant l'allocation des ressources et l'ordonnancement des tâches.
2. Les problème d'ordonnancement en ligne (*online*) pour lesquels les dates d'arrivée des tâches constituant l'application ne sont pas connues à l'avance.

Lors de la conception d'un algorithme d'allocation de ressources et d'ordonnancement de tâches, souvent un critère d'optimisation est pris en compte. Il s'agit généralement du temps d'exécution de l'application en question en utilisant une plate-forme de ressources donnée. Ce critère est aussi appelé le *makespan*. D'autres critères peuvent être optimisés tels que [81][82][83][84][85] :

1. Le temps moyen d'exécution : ce critère est utilisé lorsque plusieurs utilisateurs accèdent de manière concurrente à un ensemble de ressources.
2. Le débit : ce critère est généralement utilisé lorsque un graphe de tâches est exécuté plusieurs fois ou lorsque l'on s'intéresse à un problème d'ordonnancement en ligne.
3. Le temps d'attente moyen : ce critère est utilisé pour l'ordonnancement d'un ensemble de tâches indépendantes provenant de plusieurs utilisateurs. Il correspond au temps moyen passé dans le système, par une tâche donnée, pour son accomplissement.

La suite de ce chapitre est organisé comme suit. La section suivante est consacrée à une brève introduction aux problèmes d'allocation de ressources et d'ordonnancement de tâches ainsi qu'à l'optimisation multi-objetifs. Une classification des applications parallèles est donnée par la Section 4.3. Nous exposerons dans la Section 4.4 les différents travaux sur l'ordonnancement d'applications parallèles. La Section 4.5 conclut le chapitre.

4.2 Concepts de base

Le contexte de notre travail se situe à l'intersection de la modélisation et de la résolution des problèmes d'allocation de ressources et d'ordonnancement de tâches des processus dans le cadre du *Cloud Computing*. Nous commençons dans ce qui suit par un bref rappel de quelques généralités sur le problème d'allocation de ressources et d'ordonnancement de tâches. Par la suite, nous donnons un état de l'art non exhaustif sur l'optimisation multi-objectifs et les techniques de résolutions candidates au problème auquel nous nous intéressons. Ainsi, nous aurons tous les éléments nécessaires pour appréhender les modèles et les approches présentés dans les chapitres suivants.

Définition 1 *Un problème d'allocation de ressources et d'ordonnancement de tâches $t_i, 1 \leq i \leq n$, étant donné un ensemble de ressources $r_j, 1 \leq j \leq m$, consiste à déterminer la date $d(t_i), 1 \leq i \leq n$ de début d'exécution de chacune des tâches et à choisir les ressources qui les exécutent.*

4.2.1 Optimisation multi-objectifs

D'une manière informelle, un problème multi-objectifs peut être défini comme un problème d'optimisation dont l'objectif est déterminer une solution qui satisfait un ensemble de contraintes et optimise un vecteur de fonctions objectifs.

La difficulté d'un problème multi-objectifs est qu'il n'existe pas de solution optimale. Effectivement, généralement lorsque les objectifs pris en compte sont conflictuels, il n'existe pas de solution qui soit optimale pour tous ses objectifs. On peut donc simplement exprimer le fait qu'une solution est préférable à une autre mais on ne peut jamais affirmer qu'une solution est meilleure que toutes les autres.

Par conséquent, lors de la résolution d'un problème multi-objectifs l'objectif est de déterminer toutes les solutions satisfaisantes. C'est pour cela que les méthodes de résolution de ce type de problème sont dites méthodes d'aide à la décision car le choix final revient

au décideur (à l'utilisateur des ressources du *Cloud* dans notre cas). Formellement, un problème multi-objectifs peut être défini comme suit :

Définition 2 (Problème multi-objectifs) *un problème multi-objectifs peut être défini comme suit :*

$$\min C(x) = (c_1(x), ..., c_n(x))$$

où :

— $c_i, 1 \leq i \leq n$ *est l'ensemble des critères pris en compte.*

— $x \in X$; X *est l'ensemble des solutions réalisables (ou espace de décision).*

Les méthodes d'optimisation multi-objectifs peuvent être classées en trois catégories décrites ci-après.

Les méthodes agrégées

L'idée principale de ces méthodes repose sur le fait que l'on cherche à optimiser une fonction d'utilité donnée par :

$$U = U(c_1, ..., c_n) \tag{4.1}$$

Généralement deux types de méthodes peuvent être distinguées :

1. méthodes additives : la fonction objectif que l'on chercher à optimiser est donnée par :

$$U = \sum_{i=1}^{n} U_i(c_i)$$

2. méthodes multiplicatives : la fonction objectif à optimiser est donnée par :

$$U = \prod_{i=1}^{n} U_i(c_i)$$

Où U_i est la fonction de mise à l'échelle du critère c_i. Plusieurs techniques de résolution ont été proposées en utilisant ces méthodes et ramener ainsi le problème initial en un problème mono-objectif dont les plus utilisées sont décrits ci-après.

(a) La somme pondérée Cette technique consiste à affecter un poids à chacun des

critères pris en compte. Ces poids représentent l'importance relative que l'utilisateur attribue à chacun des critères en question. Le problème à optimiser peut alors s'exprimer comme suit :

$$\min \sum_{i=1}^{n} p_i c_i \tag{4.2}$$

Où p_i représente le poids affecté au critère c_i, avec $\sum_{i=1}^{n} p_i = 1$. Cette méthode est simple à mettre en oeuvre mais présente de nombreuses insuffisances dont :

— la difficulté à déterminer les différents poids à utiliser.
— la difficulté à définir les interactions entre les critères pris en compte.
— c'est une méthode qui souffre du problème de changement d'échelle et de compensation entre critères.
— tous les critères doivent être exprimés dans une même unité.

(b) *Goal programming* Cette méthode est appelée aussi *target vector optimisation*. L'idée de cette méthode consiste à fixer pour chacun des critères pris en compte un objectif à atteindre noté o_i (pour le critère c_i) et la fonction objectif consiste alors à minimiser la somme des écarts entre les résultats et les objectifs fixés. La fonction objectif à optimiser est donnée par l'équation suivante :

$$\min \sum_{i=1}^{n} |c_i - o_i| \tag{4.3}$$

(c) Le min-max Cette méthode est comparable à la précédente. Elle consiste à minimiser le maximum de l'écart relatif entre un critère et son objectif. Le problème à minimiser peut donc s'exprimer comme suit :

$$\min \max_{i=1}^{n} \frac{c_i - o_i}{o_i} \tag{4.4}$$

(d) Goal attainment La solution optimale en utilisant cette technique est obtenue en résolvant le problème suivant :

$\min \alpha$, tel que :

$$c_i \leq o_i + \alpha p_i \tag{4.5}$$

(e) ϵ-contrainte Cette technique consiste à optimiser un critère c_i avec la contrainte que

tous les autres critères $c_j, j \neq i$ doivent être inférieurs à une valeur ϵ_j. Le problème peut donc être exprimé comme suit :

$\min c_i$, tel que :

$$c_j \leq \epsilon_j, \forall j \neq i \tag{4.6}$$

Les méthodes Pareto

C'est Goldberg [59] qui a introduit pour la première fois l'idée d'utiliser la dominance au sens de Pareto pour résoudre les problèmes proposés dans [60]. Cette notion d'optimalité tente de respecter l'intégralité de chacun des critères de comparaison pris en compte dans le sens où il ne compare pas les valeurs des critères. En effet, dans un problème multi-objectifs on peut trouver un équilibre tel que l'on ne peut améliorer un critère sans détériorer au moins un des autres critères [61]. C'est cet équilibre que l'on appelle optimum de Pareto. Une solution est dite Pareto-optimale si elle n'est pas dominée par aucune autre solution appartenant à l'ensemble de toutes les solutions possibles. Ces points sont aussi appelés solutions efficaces ou solutions non dominées. Une définition formelle de la dominance au sens de Pareto est donnée par :

Définition 3 (Optimalité de Pareto) *Une solution réalisable $x \in X$ est dite non-dominée (ou de Pareto) s'il n'existe pas de solution $x' \in X$ tels que :*

1. $\forall i \in \{1, ..., n\}, c_i(x') \leq c_i(x)$.
2. $\exists j \in \{1, ..., n\}, c_j(x') < c_j(x)$.

La Figure 4.1 est une représentation des solutions réalisables d'un problème bi-objectifs (deux fonctions objectifs, notées $c_1(x)$ et $c_2(x)$, à minimiser). La solution x_4 domine la solution x_1, car les valeurs des deux fonctions objectifs sont plus petite en x_4 qu'en x_1 (c'est-dire $c_1(x_4) < c_1(x_1)$ et $c_2(x_4) < c_2(x_1)$). Par contre, la solution x_2 ne domine pas la solution $x4$, car $c_1(x_2) > c_1(x_4)$.

Les méthodes non agrégée et non Pareto

Les méthodes dites non agrégées et non Pareto recherchent les solution optimale en traitant séparément les fonctions objectifs prises en compte. Il exige plusieurs techniques dont :

— La technique lexicographique : les critères pris en compte sont classés par ordre d'importance. Ensuite, l'optimum est obtenu en minimisant d'abord la fonction

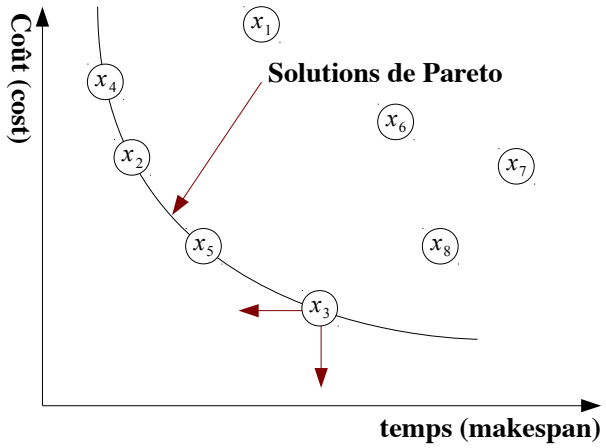

FIGURE 4.1 – Quatre solutions de Pareto (x_2, x_3, x_4, x_5) et quatre solutions dominées (x_1, x_6, x_7, x_8) du problème d'optimisation bi-objectifs (temps et coût)

objectif la plus importante (ayant un poids le plus élevé) puis la deuxième fonction objectif et ainsi de suite.

— Vector Evaluated Genetic Algorithm (VEGA) : c'est une extension d'une version classique d'un algorithme génétique classique pour résoudre un problème multi-objectifs. L'idée de cet algorithme peut être résumé comme suite : si n objectifs sont pris en compte alors k/n individus sont sélectionnés pour chacun de ces objectifs. Ainsi les n sous populations sont mélangés pour obtenir une population de taille k. Le processus de recherche de solutions optimales se termine en des opérateurs génétiques de modification (i.e. croisement et mutation).

4.3 Classification des applications parallèles

Une application parallèle est soit composée uniquement d'une seule tâche ou d'un ensemble de tâches exécutées en utilisant plusieurs ressources. Le modèle utilisé pour représenter une application parallèle dépend des tâches qui la composent et des interactions possibles entre elles.

Généralement, deux types de tâches peuvent être distinguées [36] :

1. Les tâches séquentielles : une tâche est dite séquentielle si et seulement si elle ne peut être exécutée que par une seule ressource.
2. Les tâches parallèles : une tâche est dite parallèle si et seulement si elle peut être exécutée par plusieurs ressources.

Pour ce deuxième type de tâches, on parle de parallélisme de données et il est courant d'associer des modèles d'accélération à ce type de tâches afin d'estimer leurs temps d'exécution en fonction du nombre de ressources utilisées pour les accomplir. Les tâches dites parallèles peuvent être classées en trois catégories selon leur flexibilité [37], à savoir :

1. Les tâches *rigides* où le nombre de ressources utilisées pour les exécuter est fixé à priori.
2. Les tâches *modelables* où le nombre de ressources nécessaires pour les exécuter n'est pas fixé à priori mais déterminé avant l'exécution.
3. Les tâches *malléables* où le nombre de ressources qui les exécutent peuvent évoluer au cours de l'exécution.

Dans cette thèse nous nous intéressons à l'allocation de ressources et l'ordonnancement d'applications composées de tâches rigides et de tâches séquentielles. L'ordonnancement de tâches en utilisant des ressources en nombre limité nécessite la prédiction du temps d'exécution de chacune des tâches en fonction de leurs charges de travail. Dans le cas de tâches *modelables*, cette prévision peut être obtenue en utilisant des modèles d'accélération. En effet, de nombreux modèles d'accélération ont été proposés pour la prévision des temps d'exécution des tâches parallèles. Cependant, ces modèles supposent que toutes les ressources utilisées sont homogènes et qu'elles sont une fonction qui dépend du nombre de ressources (r) allouées pour exécuter cette tâche et d'un certain nombre de paramètres liés à cette tâche. Ces paramètres sont déterminés soit en utilisant des modèles analytiques ou déduits à partir de considérations expérimentales. Une accélération $Acceleration(t, r)$ est définie comme étant le rapport entre le temps d'exécution d'une tâche parallèle t sur une seule ressource et son temps d'exécution sur r ressources. Parmi les modèles d'accélération proposés dans la littérature, nous pouvons citer :

1. La loi d'Amdalh [38] : cette loi est plus utilisée pour estimer le temps d'exécution d'une application parallèle en utilisant des ressources homogènes. Selon cette loi le temps d'exécution de la tâche t est donné par l'équation suivante :

$$ET(t, r) = (\alpha + \frac{1 - \alpha}{r})ET(t, 1) \tag{4.7}$$

   où :

   (a) α est la portion de la tâche t non parallélisable.

   (b) $ET(t, 1)$ est le temps d'exécution de la tâche t en utilisant une seule ressource.

(c) $ET(t,r)$ est l'estimation du temps d'exécution de la tâche t en utilisant r ressources.

2. Le modèle d'accélération proposé dans [39] : ce modèle propose d'exprimer le temps d'exécution d'une tâche en utilisant un ensemble de ressources en parallèle sous forme d'une fonction non linéaire. Elle dépend de deux paramètres, à savoir :

   (a) α représente le parallélisme moyen de la tâche en question.

   (b) β représente une approximation du coefficient de variation du parallélisme de la tâche à exécuter.

   Ainsi, deux types de tâches parallèles sont distinguées selon la valeur de la *variance du parallélisme* $Var = \beta(\alpha-1)^2$.

   Pour les tâches parallèles avec une faible variance (i.e. $\beta \leq 1$), l'accélération est donnée par l'équation suivante :

$$Acceleration(t,r) = \begin{cases} \frac{\alpha r}{\alpha+\beta/2(r-1)}, & 1 \leq r \leq \alpha \\ \alpha, & \alpha \geq 2\alpha - 1 \\ \frac{\alpha r}{\beta(\alpha-1/2)+n(1-\beta/2)}, & \alpha \leq r \leq 2\alpha - 1 \end{cases}$$

   Pour les tâches parallèle avec une variance élevée (i.e. $\beta > 1$), l'accélération est donnée par l'équation suivante :

$$Acceleration(t,r) = \begin{cases} \frac{\alpha r(\beta+1)}{\beta(2+\alpha-1)+\alpha}, & 1 \leq r \leq \alpha + \alpha\beta - \beta \\ \alpha, & r \geq \alpha + \alpha\beta - \beta \end{cases}$$

   Les deux modèles d'accélération présentés précédemment sont relativement simples mais ils ne prennent pas en compte les temps de communication entre les différentes ressources utilisées. Afin de prendre en compte les temps de communication, d'autres modèles ont été proposés. Parmi ces modèles, nous pouvons citer :

3. Un modèle basé sur l'idée que le temps d'exécution d'une tâche t sur r ressources est le maximum entre le temps de calcul mis pour exécuter cette tâche (i.e. $T_{exec}(t,r)$) et le temps de communication des données (i.e. $T_{comm}(T,r)$). Autrement dit, l'estimation du temps d'exécution de la tâche t en utilisant r ressources est donné par l'équation suivante :

$$ET(t,r) = \max\{T_{exec}(T,r), T_{comm}(T,r)\} \tag{4.8}$$

   Le temps de calcul est donné par l'équation suivante :

$$T_{exec}(t,r) = \frac{V(t)}{v \times r} \tag{4.9}$$

Où :

— $V(r)$ représente la quantité de calcul à effectuer exprimé en Gflop [19].

— v représente la vitesse du processeur en Gflop/s.

Le temps de communication est donné par l'équation suivante :

$$ET(t,r) = \frac{2 \times V(t) \times (r-1)}{r^2 \times B} \tag{4.10}$$

Où B est la bande passante moyenne entre les ressources qui exécutent la tâche t.

Les modèles de prévision proposés dans la littérature ne sont pas adaptés au problème traité dans cette thèse. En effet, ces modèles se focalisent sur la prédiction du nombre de ressources à utiliser pour exécuter une tâche donné (i.e. ces modèles sont adaptés au cas où l'application est constituée de tâches parallèle). Or, dans notre cas un processus est modélisé sous la forme d'un graphe de tâches séquentielles.

Afin de pallier à ces insuffisances, le deuxième modèle que l'on propose dans cette thèse consiste à prévoir le temps d'exécution d'une tâche sur une ressource donnée sachant qu'elle peut être amenée à exécuter d'autres tâches dont on n'a aucun contrôle (ces tâches arrivent de manière aléatoire dans les files d'attente des ressources utilisées).

4.3.1 Types d'applications parallèles

Deux catégories principales d'applications parallèles peuvent être distinguées selon les tâches qui les composent et les interactions entre elles, à savoir :

1. Le premier type d'applications parallèles se réfère au cas où les tâches définissant une application donnée peuvent s'exécuter indépendamment les unes des autres. Parmi ce type d'applications, il existe celles qui sont constituées de tâches parallèles et celles qui sont définies par des tâches séquentielles. Il existe un troisième type d'applications appelées *applications paramétriques (parameter sweep application)*. Dans ce dernier cas, une même application est exécutée plusieurs fois avec différentes valeurs de ses paramètres (les paramètres d'entrée de cette application peuvent être variés).

19. FLoating point Operations Per Second

2. Le deuxième type d'applications parallèles se réfère au cas où les tâches constituant l'application en question ne peuvent être exécutées de manière indépendantes et sont liées par des contraintes de précédences représentant des contraintes temporelles ou d'échange de données. Pour ce type d'applications, il existe deux catégories principales :

   (a) La première catégorie se réfère aux applications définies par des tâches qui interagissent en s'échangeant des données durant leurs exécutions. Ces applications peuvent être modélisées en utilisant des graphes non orientés appelés *Task interaction graphs (TIGs)*. Les sommets de ces graphes représentent les tâches et les arrêtes représentent les communications entre les ressources.

   (b) La deuxième catégorie se réfère aux applications définies par des tâches liées par des contraintes de précédence. Autrement dit, une tâche de ces applications ne peuvent être exécutée que si toutes les tâches qui la précèdent ont été exécutées. Ces applications sont généralement représentées par des graphes orientés sans cycle (*Directed Acyclic Graphs (DAGs)*). Les sommets de ces graphes représentent les tâches (parallèles ou séquentielles) de l'application à exécuter et les arcs définissent les relations de précédence entre ces tâches (dépendances temporelles ou de données). Dans le cas où toutes les tâches sont séquentielles, on parle d'application exploitant purement le paradigmes de parallélisme de tâches. Dans le cas contraire (i.e. l'application est composée de tâches parallèles et séquentielles), on utilise des *Temporal Task Interaction Graphs (TIGs)* pour les représenter.

4.4 État de l'art sur l'ordonnancement des processus

Nous avons expliqué dans la section précédente qu'il existe plusieurs types d'applications et de tâches qui les composent. L'un des problèmes fondamentaux qui peut dégrader les performances de ces applications est la mauvaise gestion de l'utilisation des ressources. Il est donc indispensable de développer des méthodes et des outils permettant l'utilisation efficace des ressources disponibles.

Le problème de gestion efficace des ressources réparties, en utilisant notamment une grille de calcul ou plate-forme du type *Cloud*, a été largement étudié et de nombreux algorithmes d'ordonnancement ont été proposés [177][176][175][174][173][149][148]. De plus, divers objectifs sont pris en compte selon le type de l'application et/ou des tâches à

exécuter ainsi que de la plate-forme de ressources utilisées.

Nous proposons dans ce qui suit une classification, des algorithmes existants, en trois catégories selon les types de tâches constituant l'application en question, à savoir (i) tâches indépendantes, (ii) tâches dépendantes (DAG) et (iii) DAGs s'exécutant en parallèle.

4.4.1 Ordonnancement de tâches indépendantes

Le problème d'allocation de ressources et d'ordonnancement des applications composées de tâches indépendantes est le plus étudié comparé aux deux autres types d'applications. En effet, de nombreuses études ont été proposées dans la littérature pour l'ordonnancement des tâches indépendantes. Cependant, la plupart des travaux utilisent des plates-formes homogènes.

Les algorithmes d'ordonnancement de tâches indépendantes visent généralement à minimiser le temps global d'exécution *ou le makespan* de l'application à exécuter ou à maximiser le débit (*througput*) de la plate-forme utilisée dans le cas où les tâches n'arrivent pas toutes en même temps (i.e. les tâches arrivent au fur et à mesure dans le système). Étant donné la difficulté du problème, les techniques proposées sont des algorithmes approchés.

Les algorithmes approchés ont l'avantage de garantir la qualité des solutions obtenues au pire cas par rapport à la solution optimale. Le critère utilisé est généralement un ratio qui ne doit pas être dépassé par le rapport du temps global obtenu et le temps global optimal de l'application à exécuter. En dépit de cette garantie de performances, ces algorithmes restent très peu exploitables pour l'exécution des applications concrètes. Ceci est dû au fait qu'ils sont très coûteux en terme du temps d'exécution. De plus, ils ne tiennent pas compte des temps de communication entre les différentes tâches ce qui n'est pas rare pour des applications concrètes.

Plusieurs heuristiques dites dynamiques ont été proposées dans [33] pour l'ordonnancement de tâches indépendantes et séquentielles classées en deux catégories :

1. Des heuristiques d'ordonnancement en ligne : les tâches qui arrivent dans le système sont ordonnancées et affectées à des ressources dès leurs arrivées.
2. Des heuristiques d'ordonnancement par lot (*batch scheduling*) : les tâches ne sont pas exécutées à leurs arrivées mais regroupées dans des ensembles distincts et les dates de leurs ordonnancements sont déterminées ultérieurement.

En ce qui concerne les tâches parallèles, de nombreuses heuristiques dynamiques ont été proposées et utilisées dans l'optimisation d'utilisation des ressources de nombreux

systèmes tels que OAR [1][2] [20], PBS[3] [21] , Maui[4], etc. L'objectif de ces heuristiques est essentiellement l'augmentation du taux d'utilisation des ressources et la minimisation du temps d'attente des tâches. Parmi ces heuristiques, nous pouvons citer :

1. FCFS (*First Come First Served*) ou premier arrivé premier servi. Autrement dit, les tâches sont affectées dans l'ordre de leurs arrivées dès qu'un nombre suffisant de ressources est disponible. C'est une heuristique qui garantit qu'il n y aura pas de situation de famine. Cependant, elle n'utilise aucun moyen de prévision pour déterminer le nombre de ressources nécessaires à l'exécution des tâches qui arrivent dans le système. Son inconvénient est la sous-utilisation des ressources disponibles.
2. Afin de pallier à l'insuffisance de l'heuristique FCFS, des extensions appelée « *backfiling conservatif* » ont été proposées [5][6]. Ces techniques consistent à avancer les dates d'exécution des tâches nécessitant le moins de ressources afin d'augmenter le taux d'occupation des ressources utilisées tout en assurant que des tâches plus prioritaires (i.e. les tâches étant arrivées avant dans le système) ne soient pas retardées. Ces techniques, comme l'heuristique FCFS, évitent les situations de famine (i.e. qu'une tâche n'aie jamais accès aux ressources). Des expérimentations ont montré que ces techniques augmentent considérablement le taux d'utilisation des ressources.
3. L'heuristique « *backfiling agressif* » [7] permet d'augmenter les taux d'occupation des ressources comparés aux taux d'occupation obtenus par les heuristiques dites « *backfiling conservatif* ». Cependant, elle peut causer des problèmes de famine pour des tâches nécessitant de nombreuses ressources.
4. LWF (*least work first*) : le principe de cette heuristique consiste à exécuter d'abord les tâches nécessitant le moins de ressources. Dans [6], les auteurs ont montré que l'utilisation de cette approche d'allocation et d'ordonnancement de tâches garantit un taux moyen de 70% d'occupation des ressources en considérant plusieurs plateformes.

4.4.2 Ordonnancement de tâches dépendantes

La plupart des algorithmes d'ordonnancement d'applications composées de tâches dépendantes visent à optimiser leurs temps d'exécution en se focalisant sur des applications

20. OAR est un gestionnaire de ressources pour grappe de serveurs
21. Portable Batch System

définies par des tâches liées par des contraintes de précédence (DAGs). En effet, très peu d'algorithmes ont été proposés pour le cas de tâches qui interagissent (i.e. modéliser par des TIGs) en utilisant des ressources hétérogènes [8] [9].

Le problème d'allocation de ressources et d'ordonnancement de tâches d'une application représentée avec un graphe orienté sans cycle (DAG) est un problème NP-difficile même si le graphe est composé de tâche séquentielles [10]. Par conséquent, la plupart des solutions proposées sont des heuristiques dont le critère d'optimisation pris en compte est le temps global d'exécution.

L'allocation de ressources et l'ordonnancement de graphes de tâches composées de tâches séquentielles a été largement étudié et plusieurs algorithmes heuristiques ont été proposés. Ces algorithmes peuvent être classés en trois catégories :

1. *Les algorithmes d'ordonnancement de listes* [11][12][13]. Ces algorithmes se déroulent en deux phases. La première phase consiste à calculer la priorité des tâches constituant l'application en question. Ainsi des listes de tâches sont constituées. Le calcul de la priorité d'une tâche dépend des contraintes de précédence qui la lient avec les autres tâches à exécuter. Par conséquent, la priorité de cette tâche est inférieure à la priorité de toutes les tâches qui doivent être exécutes avant elle. La deuxième phase de ces algorithmes consiste à choisir une ressource pour chacune des tâches dans l'ordre de priorité tout en minimisant une fonction de récompense préalablement définie (par exemple, la date de fin d'une tâche).

   Ces heuristiques ont l'avantage d'être peu complexes comparées aux autres types d'heuristiques et produisent des résultats similaires.

2. *Les algorithmes d'ordonnancement de clustering* [14][15]. Ces algorithmes se déroulent en deux phases. La première phase consiste à regrouper les tâches dans des groupes (*cluster*). La deuxième phase consiste à affecter ces tâches aux ressources disponibles. La contrainte à respecter est que toutes les tâches appartenant à un même groupe soient exécutées par une même ressource.

3. *Les algorithmes d'ordonnancement de duplication de tâches* [16][17][18][19]. Afin de réduire les temps de communication dûs aux échanges de données, certains algorithmes exécutent une même tâche sur plusieurs ressources. Un des inconvénients de ces algorithmes est qu'ils sont plus coûteux que ceux basés sur la construction de listes ou de groupes de tâches.

4. *Les algorithmes d'ordonnancement méta-heuristiques* D'autres algorithmes[26] [27] [28] qualifiés de méta-heuristiques ont été proposés pour l'ordonnancement de

graphes de tâches séquentielles. Ces heuristiques sont généralement basées sur les méthodes de *Recherche Tabou*, *RecuitSimulé* et *Algorithmes Génétiques*. Ces derniers restent les plus utilisés et produisent de meilleurs résultats que les autres méthodes. Cependant, ils sont plus coûteux en terme de temps d'exécution (complexité).

Quant au problème d'allocation et d'ordonnancement d'applications composées de tâches parallèles, plusieurs algorithmes on été proposés en utilisant des ressources homogènes [20][21][22][23][24][25]. Ces algorithmes sont composés de deux phases : (i) la phase d'allocation des ressources et (ii) la phase de placement des tâches. L'objectif de la première phase est de déterminer le nombre de ressources à utiliser pour exécuter chacune des tâches. Une fois le nombre de ressources fixé pour toutes les tâches, la deuxième phase consiste à déterminer l'ordre d'exécution des tâches sur chacune des ressources.

4.4.3 Ordonnancement d'applications concurrentes

Le critère d'optimisation pris en compte par les algorithmes visant à ordonnancer un ensemble d'applications partageant les mêmes ressources est le *makespan* ou le débit de la plateforme de ressources utilisées. C'est un problème difficile du fait qu'il faut assurer l'équité entre les différentes applications exécutées en parallèle. Un ordonnancement non équitable peut entrainer au pire des cas des situations de famine. Autrement dit, un sous ensemble des applications concernées n'auront jamais accès aux ressources disponibles (certaines tâches seront indéfiniment retardées). La notion d'équité a été utilisée et interprétée sous différentes formes [29] [30] [31] [32]. Il convient de souligner que tous ces algorithmes considèrent le cas où les applications sont constituées de tâches séquentielles.

Dans [34], les auteurs ont proposé un algorithme appelé *Steady-state* pour l'ordonnancement d'un ensemble d'instances décrites par des DAGs. Cet algorithme est utilisé en régime permanent introduit dans [35]. Au moyen d'un programme linéaire en nombres rationnels et asymptotiquement (le nombre de graphe tend vers l'infini), cet algorithme permet d'obtenir la solution optimale du problème d'ordonnancement d'un ensemble de graphes. Une extension de cet premier algorithme a été proposé dans [36] pour ordonnancer un ensemble borné de graphes de tâches.

4.5 Synthèse

Nous avons survolé, dans ce chapitre, le domaine de l'allocation des ressources et de l'ordonnancement de tâches. Plus précisément, nous avons présenté une brève introduction à l'allocation des ressources et à l'ordonnancement des tâches ainsi que les techniques candidates à la résolution de ce type de problème. Nous avons aussi mis l'accent sur l'importance d'utiliser une approche multi-objectifs pour apporter des réponses satisfaisantes au problème auquel nous nous intéressons. Les limites des études que nous avons examinées et qui constituent les points de départ à notre travail peuvent être résumées par les points suivants :

1. Les algorithmes existants pour l'allocation de ressources et d'ordonnancement des tâches des processus ne sont pas adaptés dans le cas de l'utilisation d'une plateforme de *Cloud Computing*. En effet, ces algorithmes ne prennent pas en compte les spécificités de ce dernier tels que « l'illusion de ressources infinies » et son modèle économique basé sur la consommation réelle.
2. Lorsque plusieurs critères de qualité de service sont pris en compte simultanément, l'approche utilisée consiste à utiliser une fonction d'agrégation du type somme pondérée. Cette approche souffre de nombreuses insuffisances dont le problème de changement d'échelle et le problème de compensation entre les critères pris en compte.
3. La plupart des travaux de recherche sur l'ordonnancement de plusieurs graphes de tâches ne prennent en compte qu'un seul critère d'optimisation (*makespan*) et les graphes exécutés en parallèle sont tous identiques [145][146][147].
4. La plupart des algorithmes utilisent des ressources homogènes.

Pour dépasser ces limites, nous abordons dans la suite de ce rapport le problème d'allocation de ressources et d'ordonnancement de tâches de processus sous trois aspects complémentaires, à savoir :

1. Proposition d'approches d'allocation de ressources et d'ordonnancement de processus scientifiques (toutes les tâches sont automatisées).
2. Proposition d'approches d'allocation de ressources et d'ordonnancement de tâches d'une instance d'un processus métier (un sous ensemble de tâche nécessite l'intervention de ressources humaines pour les accomplir).
3. Généralisation des deux premières études au cas où plusieurs instances d'un modèle de processus s'exécutent en concurrence.

Dans la suite de ce document, nous proposons des solutions essentiellement aux limitations énumérées ci-dessus. La suite est organisée comme suit.

Chapitre 5 Le succès commercial du *Cloud Computing* peut être compromis par de nombreux problèmes tels que les problèmes de sécurité et le manque d'outils permettant aux utilisateurs de mieux choisir les ressources à utiliser pour exécuter leurs applications. Nous nous intéressons dans ce chapitre au problème du choix de la meilleure ressources afin de minimiser deux critères de qualité de service (le temps d'exécution et le coût d'exécution).

Chapitre 6 Nous présentons dans ce chapitre une extension des approches proposées dans le chapitre 5. Plus précisément, nous proposons des approches d'allocation de ressources et d'ordonnancement de tâches en prennent en compte deux types de ressources : (i) ressources humaines et (ii) machines virtuelles. L'introduction de ce type de ressources (notamment humaines) engendre le problème lié à la gestion de l'ordonnancement au niveau de chaque file d'attente modélisant une ressource donnée. Et afin de prendre en compte les phénomènes aléatoires sous-jacents à l'utilisation des ressources humaines, nous proposons une heuristique basée sur le lissage exponentielle.

Chapitre 7 L'objectif de ce chapitre est de proposer des stratégies d'allocation de ressources et d'ordonnancement de tâches de plusieurs instances d'un modèle de processus qui s'exécutent en parallèle. En plus des deux fonctions objectifs prises en compte dans les deux chapitres précédents, nous proposons un troisième critère permettant d'assurer un partage équitable des ressources utilisées.

Chapitre 5

Optimisation bi-objectifs d'exécution des processus scientifiques

Sommaire

5.1 Introduction

Le *Cloud Computing* a suscité un engouement pour l'exécution des processus. Il est basé sur la notion de virtualisation. Par conséquent, les utilisateurs peuvent exécuter leurs procesus en utilisant des machines virtuelles. Le modèle économique de ce paradigme est basé sur le paiement de la consommation réelle des utilisateurs. Dans ce contexte, les utilisateurs ne seront pas amenés à s'occuper de l'infrastructure mais uniquement des services qu'ils utilisent. De nombreuses entreprises l'ont rapidement adopté afin produire vite et à moindre coût. Cependant, malgré les nombreux avantages avérés d'utiliser des ressources du *Cloud* pour exécuter des workflows, cette infrastructure reste confrontée à un problème majeur qui peut compromettre son succès commercial. Il s'agit de l'absence d'outils permettant à l'utilisateur de choisir un fournisseur parmi les offres multiples disponibles sur le marché, pour exécuter ses processus. Il est, par conséquent, indispensable de développer des méthodes et des outils afin de satisfaire au mieux les utilisateurs.

Généralement, deux types de processus (*workflows*) peuvent être distingués, à savoir : (1) processus scientifiques et (2) processus métiers. Nous nous intéressons, dans ce chapitre, au premier type de processus.

Un processus scientifique est communément représenté par un graphe sans cycle où les nœuds représentent les tâches de l'application et les arcs les dépendances entres elles. Nous distinguons deux types de dépendances : (1) dépendances temporelles et (2) dépendances de données.

Par ailleurs, il a été démontré que le problème d'allocation et d'ordonnancement des tâches d'un workflow est un problème NP-complet. Il est donc plus approprié de proposer des heuristiques au lieu de méthodes exactes pour ce type de problème.

Plusieurs études se sont intéressées au problème d'ordonnancement d'un graphe acyclique de tâches en utilisant des ressources hétérogènes notamment [65][175][176][177]. Ces études présentent, cependant, de nombreuses insuffisantes qui les rendent inexploitables lorsque l'on utilise la plate-forme du *Cloud Computing*. Ces insuffisances se déclinent selon les points suivants :

1. Le nombre de ressources mises à la disposition des utilisateurs est supposé fini. Or, « l'illusion du nombre infini de ressources (élasticité) », comme nous l'avons mentionné précédemment, est l'une des caractéristiques les plus importante du paradigme *Cloud Computing*.
2. Le seul critère d'optimisation pris en compte est généralement le temps d'exécution.

Cependant, le modèle économique du *Cloud Computing* est basé sur une facturation de la consommation réelle. Le coût d'exécution est donc un deuxième objectif qui doit être considéré.

Pour pallier à ces insuffisances, nous proposons dans ce chapitre des approches dont le but est d'optimiser l'exécution des processus scientifiques en utilisant des ressources via la plate-forme du *Cloud Computing*. De plus, nous prenons en compte deux critères conflictuels, à savoir le temps d'exécution et le coût engendré par l'utilisation d'un ensemble de ressources.

Le problème traité dans ce chapitre (allocation de ressources et ordonnancement de graphe de tâches) est connu pour son appartenance à la classe des problèmes dits NP-complet. Par conséquent, nous avons développé des heuristiques pour évaluer la qualité des solutions que l'on obtient par nos approches. Plus précisément, nous avons défini deux bornes inférieures des deux critères d'optimisation pris en compte.

Les apports de ce chapitre sont :

1. Proposition d'une formalisation du problème d'allocation de ressources et d'ordonnancement de tâches d'un processus scientifique lorsque l'on utilise la plate-forme du *Cloud Computing*.
2. Élaboration d'une première approche bi-objectifs d'allocation de ressources et d'exécution des tâche d'un processus scientifique. Cette approche est dirigée par la fonction objectif du coût d'exécution. Cependant, la valeur de la fonction objectif du temps d'exécution est calculée pour chacune des solutions obtenues.
3. Proposition d'une deuxième approche bi-objectifs. Elle est dirigée par la fonction objectif du temps d'exécution. De même que pour la précédente première approche, la valeur de la fonction objectif du coût d'exécution est calculée pour chacune des solution obtenue.
4. Proposition d'une troisième approche dont le but est de ne retenir que les solutions dites non dominées obtenues par les deux premières approches.
5. Définition de deux bornes inférieures des valeurs des deux critères d'optimisation pris en compte.

Le reste de ce chapitre est organisé comme suit. Dans la section suivante, nous proposons une formulation du problème d'allocation de ressources et d'ordonnancement des tâches constituant un workflow scientifique. La Section 5.4 est consacrée à la présentation des trois approches complémentaires pour traiter du problème d'ordonnancement des

workflows scientifiques. Étant donnée l'appartenance du problème traité dans ce chapitre à la classe NP-complet, et afin d'évaluer les performances des résultats obtenus par nos algorithmes, nous proposons dans la Section 5.4 deux bornes inférieures pour les deux critères pris en compte, à savoir le temps global d'exécution et le coût engendré par l'utilisation d'un ensemble de ressources. La section 5.6 est dédiée à la présentation d'un récapitulatif des résultats numériques obtenues. La section 5.7 conclut le chapitre.

5.2 Formulation du problème d'allocation et d'ordonnancement d'un processus scientifique

Dans ce qui suit, nous commençons par présenter le problème auquel nous nous sommes intéressés puis nous décrivons les fonctions objectifs que nous avons prises en compte.

Rappelons que l'objectif principal de ce chapitre est la modélisation et la résolution du problème d'allocation de ressources et d'ordonnancement de tâches d'un workflow scientifique dans le cadre du *Cloud Computing*. Un workflow scientifique est communément représenté par un graphe orienté sans cycle (DAG, *Directe Acyclic Graph*). Formellement, il est défini comme suit :

Définition 4 (Workflow) *Un workflow scientifique est représenté par un graphe orienté et sans cycle (acyclique) noté $G = (T, E)$, avec :*

- *$T = \{t_1, ..., t_n\}$ est l'ensemble fini des tâches qui le compose*
- *E est l'ensemble de ses arcs représentant les contraintes de précédence et/ou de données entre les tâches T.*
- *Un arc (t_i, t_j) du graphe G existe si et seulement s'il existe une contrainte temporelle et /ou de données entre les tâches t_i et t_j. Si une contrainte temporelle existe entre ces deux tâches alors t_j ne peut être exécutée que lorsque la tâche t_i a été exécutée. Lorsque il s'agit de contrainte de données, alors celles qui sont générées à la fin de l'exécution de la tâche t_i sont utilisées par la tâche t_j pour son exécution.*

La tâche t_i est appelée le parent immédiat de la tâche t_j. Cette dernière est appelée fils immédiat de la tâche t_i. Notons que le fils immédiat d'un ensemble de tâches ne peut être exécuté que lorsque tous ses parents immédiats ont été exécutés. Dans un graphe donné une tâche sans précédences (parents) représente le début du processus et est notée t_{input}. Une tâche sans successeurs immédiats représente la fin du processus en question et est notée t_{exit}.

Les approches que l'on propose sont appliquées sur des graphes de tâches ayant une seule tâche d'entrée (c'est-dire une seule tâche t_{input} sans prédécesseur) et une tâche de fin (t_{exit}). Si un processus est instancié en exécutant plus d'une tâche et qu'il se termine en exécutant en parallèle plusieurs tâches, alors une tâche d'entrée et une tâche de sortie sont ajoutées au graphe initial avec un temps (temps d'exécution et temps de transfert) et un coût (coût d'exécution et coût de transfert) d'exécution nuls. Notons que l'ajout des ces deux tâches n'affecte pas le résultat des approches que l'on propose. Ainsi, cette modification permet d'obtenir un graphe de tâches sans cycle avec une seule tâche d'entrée (t_{input}) et une seule tâche de fin (c'est-à-dire une seule tâche t_{exit} sans successeur).

Par ailleurs, une tâche est une et indivisible dans le sens qu'elle ne peut être divisée en sous-tâches et ainsi être exécutée sur différentes machines virtuelles. La Figure 5.1 représente un exemple d'un workflow constitué de dix tâches notées $t_1, t_2, ..., t_{10}$ représentées par des sommets et les dépendances entre elles sont représentées sous forme d'arcs. La tâche initiale (t_1) peut avoir des données en entrée, notées *input.file*, pour commencer son exécution (déclencher l'exécution du processus en question) et la tâche finale peut produire des données à la fin de son exécution notées *output.file*. Les tâches consomment et produisent des données. Ces données sont par la suite échangées entre les machines virtuelles utilisées.

Définition 5 (Données échangées) *Les données échangées entre les différentes machines virtuelles résultant de l'exécution des tâches constituant le workflow en question sont représentées par une matrice notée D de dimension $n \times n$ où :*

- *$data[i, j]$ représente les données transmises de la machine virtuelle qui exécute la tâche t_i ($VM(t_i)$) à la machine virtuelle qui exécute la tâche t_j ($VM(t_j)$).*
- *$data[i, i] = 0$. Ce qui signifie que les données générées à l'exécution de la tâche t_i ne sont pas utilisées pour l'exécuter, par exemple, une deuxième fois. Une tâche donnée du processus est exécutée une et une seule fois.*

Pour exécuter une telle application, nous disposons d'un ensemble de machines virtuelles regroupées dans des catégories. Chaque catégorie possède ses propres caractéristiques telles que le débit, le temps d'exécution, etc. Le nombre de catégories à la disposition des utilisateurs est en nombre fini. Cependant, une machine virtuelle appartenant à une catégorie donnée peut être instanciée autant de fois que nécessaire. Par conséquent, le nombre de machines virtuelles mises à la disposition des utilisateurs est en nombre infini. Ceci traduit la caractéristique d'« illusion de ressources infinies » introduites pré-

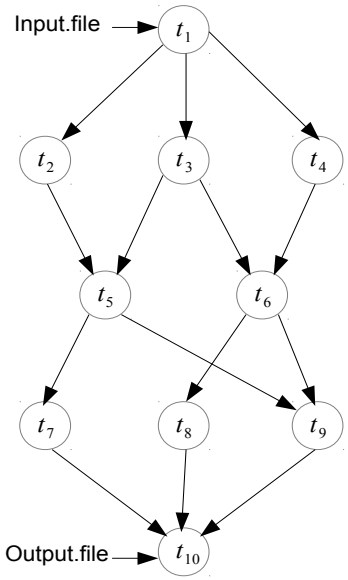

FIGURE 5.1 – Un exemple d'un processus scientifique (DAG)

cédemment. Ces machines virtuelles sont représentées sous forme d'un graphe orienté. Formellement, les ressources mises à la disposition des utilisateurs sont définies par :

Définition 6 (Graphe de ressources) *Les catégories des machines virtuelles mises à la disposition des utilisateurs sont représentées sous forme d'un graphe orienté noté* $RG = (VM, E)$*, avec :*

- $VM = \{VM_1, ..., VM_m\}$ *est l'ensemble des catégories des machines virtuelles définissant une image de chaque catégorie et leur localisation au niveau des différents centre de données. Quand il n'y a pas d'ambiguité nous utilisons le terme machine virtuelle au lieu de catégorie de machine virtuelle pour désigner une image de cette dernière.*
- E *représente l'ensemble des arcs entre les différentes catégories de machines virtuelles. Chaque arc* (VM_i, VM_j) *représente le lien entre les deux machines virtuelles* VM_i *et* VM_j*.*

Définition 7 (Débit entre machines virtuelles) *Soit* B *une matrice de dimension* $m \times m$ *représentant les débits entre les différentes machines virtuelles de l'ensemble* VM*, avec :*

- $B[i, j]$ *est le débit entre la machine virtuelle* VM_i *et la machine virtuelle* VM_j*. Notons que cette matrice n'est pas forcément symétrique. En effet, le débit entre*

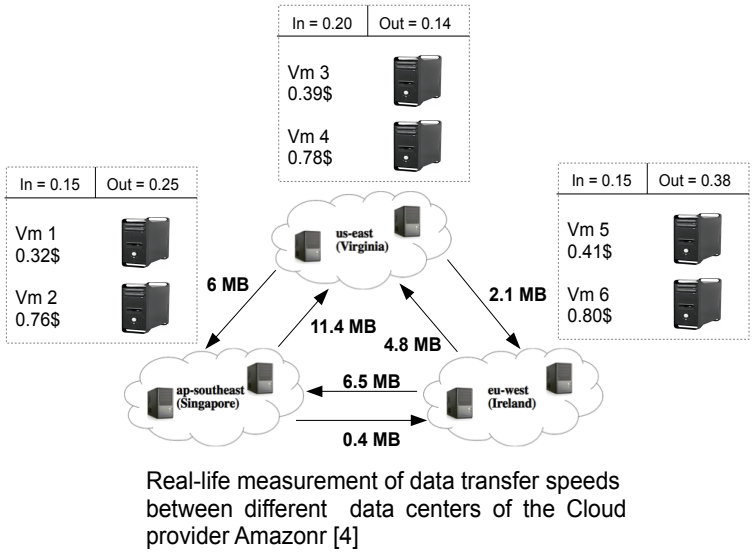

FIGURE 5.2 – Un exemple concret des débits entre différentes catégories de machines virtuelles du fournisseur d'accès Amazon

VM_i et VM_j n'est pas forcément le même qu'entre celui qui est assuré entre VM_j et VM_i.

— *$B[i,j] \longrightarrow \infty$ signifie qu'il n' y a pas de transfert de données entre les machines virtuelles VM_i et VM_j. C'est le cas par exemple actuellement lorsque deux machines virtuelles sont instanciées sur le même site (localisation) du fournisseur de services Amazon. Remarquons que c'est toujours le cas lorsque deux tâches t_i et t_j sont exécutées sur une même machine virtuelle. Autrement dit, $B[i,j] \longrightarrow \infty$.*

La Figure 5.2 est un exemple concret des débits entre les catégories de machines virtuelles du fournisseur de service Amazon, entre trois centres de données *us-east (Vergenai)*, *eu-ouest (Irlande)* et *ap-south-east (Singapour)*.

Soit $VM(t_j)$ la machine virtuelle sur laquelle la tâche t_j est exécutée. Le temps de transfert entre les machines virtuelles est donné par la définition suivante :

Définition 8 (Temps de transfert) *Le temps de transfert entre deux machines virtuelles exécutant les tâches t_i et t_j, c'est-dire $VM(t_i)$ et $VM(t_j)$, est déterminé en fonction de la quantité des données transférées et du débit entre ces machines. Il est donné par :*

$$TT(VM(t_i), VM(t_j)) = \frac{data[i,j]}{B[VM(t_i), VM(t_j)]}$$

Définition 9 (Temps d'exécution) *Soit ET une matrice de dimension $n \times m$ où :*

- *$ET(t_i, VM_j)$ est l'estimation du temps d'exécution de la tâche t_i sur la machine virtuelle VM_j.*
- *Le temps d'exécution réel d'une tâche sur une machine virtuelle donnée est en fonction de la quantité de données générées avant son exécution.*

Un exemple d'une matrice d'estimation des temps d'exécution des tâches du processus de la Figure 5.1 par les machines virtuelles de la Figure 5.2 est donné par la Figure 5.3.

Définition 10 (Coût d'exécution unitaire) *Soit UEC un vecteur de dimension m représentant le coût unitaire engendré par l'utilisation des machines virtuelles mises à la disposition des utilisateurs. $UEC(VM_j)$ représente le coût par unité de temps engendré par l'utilisation de la machine virtuelle VM_j.*

Définition 11 (Coût d'exécution) *Soit EC une matrice de dimension $n \times m$ où un élément de cette matrice noté $EC(t_i, VM_j)$ représente le coût d'exécution engendré par l'utilisation de la machine virtuelle VM_j pour exécuter la tâche t_i. Cette quantité est donné par la formule suivante :*

$$EC(t_i, VM_j) = ET(t_i, VM_j) \times UEC(VM_j)$$

Définition 12 (Coût de transfert) *Le coût de transfert entre deux machines virtuelles VM_i et VM_j sur lesquelles sont exécutées respectivement les tâches t_i et t_j (i.e. $VM(t_i)$ et $VM(t_j)$) est donné par :*

$$TC(VM(t_i), VM(t_j)) = data[i, j] \times (C_{out}(VM(t_i)) + C_{in}(VM(t_j)))$$

Où :

- *$C_{out}(VM(t_i))$ représente le coût dû au transfert de données généré par l'exécution de la tâche t_i sur la machine virtuelle $VM(t_i)$.*
- *$C_{in}(VM(t_j))$ représente le coût dû à la réception des données nécessaires pour exécuter la tâche t_j sur la machine virtuelle $VM(t_j)$.*
- *Ces deux derniers coûts sont différents et sont définis en fonction de la localisation des machines virtuelles utilisées (centres de données). C'est le cas par exemple actuellement du fournisseur d'accès Amazon.*

VM / Task	VM_1	VM_2	VM_3	VM_4	VM_5	VM_6
t_1	13	19	18	11	8	16
t_2	24	14	13	19	21	10
t_3	24	8	14	17	18	11
t_4	21	70	16	18	21	19
t_5	8	5	14	10	18	6
t_6	18	12	11	20	18	10
t_7	7	15	11	21	20	19
t_8	8	17	9	14	13	24
t_9	12	13	21	6	18	14
t_{10}	25	11	25	11	13	8

FIGURE 5.3 – Un exemple d'une matrice d'estimation des temps d'exécution des tâches du processus de la Figure 5.1 par les machines virtuelles de la Figure 5.2

Les paramètres, décrits ci-dessus, que l'on propose de prendre en compte dans la définition de notre modèle sont généralement le résultat d'une négociation entre l'utilisateur et les fournisseurs de services. Cette négociation est sanctionnée par l'élaboration d'un document appelé SLA (Service Level Agreement) [158][159][160].

Après avoir décrit les différents paramètres pris en compte dans notre modèle d'allocation de ressources et d'ordonnancement de tâches de workflows scientifiques, nous décrivons dans ce qui suit les deux fonctions objectifs prises en compte. Notons d'emblée que ces deux objectifs sont conflictuels dans le sens où l'augmentation de l'un n'engendre pas forcément l'augmentation de l'autre.

5.2.1 La fonction objectif du temps global d'exécution

Le problème traité dans ce chapitre est à la fois un problème d'allocation de ressources et d'ordonnancement de tâches d'un workflow scientifique. Avant d'énoncer la fonction objectif du temps global d'exécution d'un processus donné, nous introduisons les deux paramètres suivants définis pour un ordonnancement partiel donné. Un ordonnancement partiel se réfère à la situation où uniquement un sous ensemble de tâches est affecté aux ressources. Ces deux paramètres sont définis comme suit :

- Le date de début au plus tôt de l'exécution d'une tâche t_j noté $ST(t_j)$ (*earliest start time*).
- La date de fin au plus tôt de l'exécution d'une tâche t_j noté $FT(t_j)$ (*earliest finish time*).

Pour calculer ces deux paramètres pour une tâche t_i, nous n'avons pas besoin effectivement de connaître les ressources qui exécutent toutes les tâches du processus en question mais uniquement les ressources utilisées pour exécuter les tâches qui précèdent la tâche t_j. C'est pour cela que l'on parle d'`ordonnancement partiel`. Plus précisément, ces deux paramètres sont calculés comme suit. Pour la première tâche du workflow à exécuter (i.e. t_{input}) :

$$ST(t_{input}) = 0 \tag{5.1}$$

$$FT(t_{input}) = ST(T_{input}) + ET(t_{input}, VM(t_{input})) \tag{5.2}$$

Pour les autres tâches du graphe, les valeurs du ST et FT sont calculées de manière récursive, commençant par la tâche initiale comme le montre respectivement les équations 5.4 et 5.3.

Pour calculer la valeur de la date de fin au plus tôt FT d'une tâche t_j, toutes les tâches qui la précédent immédiatement doivent être affectées et ordonnancées en considèrerant le temps de transfert. Les valeurs des dates de fin et de début au plus tôt d'une tâche t_j sont données respectivement par l'équation 5.3 et l'équation 5.4.

$$FT(t_j) = ST(t_j) + ET(t_j, VM(t_j)) \tag{5.3}$$

$$ST(t_j) = \max_{t_p \in pred(t_j)} \{FT(t_p) + TT(VM(t_p), VM(t_j))\} \tag{5.4}$$

avec $pred(t_j)$ l'ensemble des prédécesseurs de la tâche t_j.

Après que toutes les tâches du graphe aient été ordonnancées, le temps global d'exécution (*schedule length*) sera le temps de fin au plus tôt de la dernière tâche du processus en question (c'est-dire t_{output}). Nous appelons dans la suite de ce chapitre ce temps global d'exécution *makespan*. Il est donné par l'équation suivante :

$$makespan = FT(t_{exit}) \tag{5.5}$$

Dans le cas où les transformations, décrites précédemment, permettant de se ramener à un graphe sans cycle avec une seule tâche représentant la fin du processus à exécuter, la fonction objectif du temps global d'exécution peut être définie comme suit :

$$makespan = \max_{i \in \{1,...,k\}} \{FT(t^i_{exit})\} \tag{5.6}$$

avec, $t^i_{exit}, i \in \{1, ..., k\}$ sont les tâches à exécuter de manière parallèle pour terminer l'exécution du workflow en question.

5.2.2 La fonction objectif du coût global d'exécution

Nous décrivons dans ce qui suit la fonction objectif du coût global engendré par l'utilisation d'un ensemble de machines virtuelles pour exécuter un workflow scientifique donné. Ce coût global est composé de deux quantités, à savoir :

— Le coût d'exécution des tâches constituant le workflow à exécuter.

— Le coût de transfert dû aux échanges de données entre les différentes machines virtuelles utilisées.

Ainsi, la fonction du coût global d'exécution, notée *cost* peut être définie comme étant la somme de ces deux quantités pour toutes les tâches à exécuter. Elle est donnée par l'équation suivante :

$$cost = \sum_{j=1}^{n} \left\{ EC(VM(t_j)) + \sum_{p \in pred(t_j)} TC(VM(t_j), VM(t_p)) \right\} \tag{5.7}$$

La fonction coût global d'exécution consiste à déterminer une allocation des ressources pour exécuter les tâches d'un workflow scientifique de façon à ce que sa valeur soit minimale.

5.3 Approches candidates pour la résolution du problème traité

Le problème auquel nous nous sommes intéressés dans ce chapitre consiste à exécuter les tâches d'un workflow scientifique tout en prenant en compte deux critères de qualité de service, à savoir le temps global d'exécution et le coût global d'exécution. Ce problème ainsi défini peut être abordé de plusieurs façons :

1. En le transformant en un problème *mono-critère*. Pour cela deux poids sont attribués aux deux fonctions objectifs prises en compte. Soient p_1 et p_2 les deux poids attribués respectivement aux fonctions temps global et coût global d'exécution. Le problème revient alors à minimiser la somme pondérée des ces deux fonctions :
   $\min\{p_1 \times makespan + p_2 \times cost\}$, sous les contraintes :

$$\begin{cases} \text{contraintes de précédence du graphe de tâches (DAG)} & (1) \\ \text{prise en compte des paramètres définis précédemment} & (2) \end{cases}$$

2. En utilisant une approche dite ϵ-*contrainte*. Cette approche consiste à transformer le problème initial en définissant une des deux fonctions objectifs comme étant une contrainte dont la valeur ne dépasse pas un certain seuil fixé à l'avance. Formellement et sans perte de généralité, soit la fonction coût global d'exécution ($cost$) à transformer en une contrainte dont la valeur ne dépasse pas un seuil noté ϵ. Le problème peut être ainsi formulé somme suit :
   $\min\{makespan\}$, sous les contraintes :

$$\begin{cases} cost \leq \epsilon & (1) \\ \text{contraintes de précédence du graphe de tâches (DAG)} & (2) \\ \text{prise en compte des paramètres définis précédemment} & (3) \end{cases}$$

3. En utilisant une approche bi-objectifs. Le problème initial n'est donc pas transformé dans le sens où les deux fonctions objectifs sont prises en compte simultanément. Formellement, le problème peut être défini comme suit :
   $\min\{makespan\}$ *et* $\min\{cost\}$, sous les contraintes :

$$\begin{cases} \text{contraintes de précédence du graphe de tâches (DAG)} & (2) \\ \text{prise en compte des paramètres définis précédemment} & (3) \end{cases}$$

Étant donné qu'aucun des critères ne domine l'autre (voir Section 5.6.2), nous avons opté pour la dernière approche décrite ci-dessus. Pour cela, nous proposons dans ce qui suit trois approches complémentaires :

1. La première approche est dirigée par la fonction objectif coût global d'exécution. Notons que pour chaque solution obtenue (ordonnancement possible) la valeur de la fonction objectif du temps global d'exécution correspondant est calculée.
2. La deuxième approche est basée sur la fonction objectif temps global d'exécution. De même que pour la première approche, pour chaque solution obtenue la valeur de la fonction coût global d'exécution correspondant est calculée.
3. La troisième approche combine les deux premières approches dans le but de sélectionner uniquement les solutions dites non dominées (Pareto).

5.4 Approches bi-critères pour l'exécution d'un processus scientifique

Comme mentionné précédemment, nous proposons dans cette section trois approches complémentaires pour l'allocation des ressources et l'ordonnancement des tâches d'un processus scientifique donné en utilisant la plateforme du *Cloud Computing*. Rappelons que l'on dispose d'autant de machines virtuelles que nécessaire pour l'exécution du processus en question. Autrement dit, contrairement par exemple aux grilles de calcul, le nombre de ressources est infini et ses utilisateurs ne payent que pour ce qu'ils consomment réellement. C'est pour cela principalement que les approches proposées, pour l'allocation de ressources et l'ordonnancement des workflows, dans la cadre des grilles de calcul ne peuvent être directement appliquées lorsque l'on utilise des ressources déployées dans une plate-forme du *Cloud*.

Plus précisément, la première approche, basée sur la fonction objectif du coût global d'exécution, tente de minimiser à la fois les coûts d'exécution des tâches, d'un processus scientifique donné, et les coûts de communication engendrés par les transferts de données entre les machines virtuelles utilisées. Il est important de noter que pour chaque solution (affectation) obtenue le temps global d'exécution correspondant est calculé. La deuxième approche, possède les mêmes étapes que la première approche, mais elle est basée sur la fonction objectif du temps global d'exécution. De la même façon que dans la première approche, pour chaque solution obtenue le coût global engendré par l'utilisation d'un ensemble de ressources est calculé. Le schéma général des ces deux approches est donné par l'Algorithme 1. Finalement, nous proposons une troisième approche dont l'objectif est de ne retenir que les solutions dites non-dominées (ou de Pareto). Autrement dit, à partir

des solutions obtenues par les deux premières approches uniquement celles de Pareto sont retenues. Nous décrivons en détail dans ce qui suit ces trois approches.

Algorithm 1 étapes des deux approches proposées

1: regrouper les tâches dans des ensembles distincts ;
2: définir des stratégies d'allocation de ressources ;
3: sélectionner les solutions non dominées ;

5.4.1 Approche dirigée par la fonction du coût global d'exécution (*global execution cost driven-approach*)

Dans cette approche, le critère principal d'optimisation est le coût engendré par l'utilisation d'un ensemble de ressources (machines virtuelles). Pour cela, nous distinguons comme introduit précédemment, deux types de coûts à savoir (i) le coût d'exécution et (ii) le coût de transfert dû aux échanges de données - générés à la suite d'exécution des tâches en question - entre les différentes ressources utilisées. Cependant pour chaque solution faisable obtenue, en utilisant l'une des stratégies décrites ci-après, le temps d'exécution global correspondant est calculé.

Rappelons que l'objectif de notre étude est d'affecter une machine virtuelle pour l'exécution de chacune des tâches constituant l'application en question tout en respectant un ensemble de contraintes de précédences qui les lie. L'approche basée sur le coût consiste en l'allocation de machines virtuelles, supposées en nombre infini, pour l'exécution d'un ensemble de tâches dépendantes. Comme introduit précédemment, à chaque instant un utilisateur peut solliciter et obtenir, en principe, autant de ressources que nécessaires pour l'exécution de son processus. Cette approche est constituée de trois phases principales, à savoir :

— (i) phase de regroupement des tâches,
— (ii) phase d'allocation de ressources,
— et (iii) phase de sélection des solutions non dominées.

Un aperçu de l' approche basée sur le coût est donné par l'Algorithme 2. Nous détaillons dans ce qui suit ces trois phases.

1. Phase de regroupement des tâches

Dans le but de regrouper les tâches dans des ensembles distincts, le graphe modélisant l'application concernée est parcouru de haut en bas et les tâches sont affectées, au fur et à

Algorithm 2 Approche basée sur le coût global d'exécution

1: read the DAG, the RG and associated attributes values ;
2: sort tasks at each level by traversing the DAG in a top-down fashion ; // let L be the set of levels and l_k the tasks // belonging to the level k
3: $k \leftarrow 1$; // first level
4: **while** $(k \leq L)$ **do**
5: for all tasks $t_i \in l_k$, compute $VM(t_i)$ using equation 5.12
// assign task t_i to the virtual machine $VM(t_i)$
// that minimizes the execution cost
$mincost[k, t_i] \leftarrow VM(t_i)$; // $mincost$ is a $L \times m$ matrix
// where line k corresponds to the assignment of all tasks
// obtained starting by the tasks assignement belonging to
// this level
6: $h \leftarrow k + 1$; // compute $VM(t_i)$ for all tasks that belong // to levels $h > k$
7: **while** $(h \leq L)$ **do**
8: for all tasks $t_i \in l_h$, compute $VM(t_i)$ using equation 5.9
$mincost[k, t_i] \leftarrow VM(t_i)$;
9: $h \leftarrow h + 1$
10: **end while**
11: $h \leftarrow k - 1$; // compute $VM(t_i)$ for all tasks that belong // to levels $h < k$
12: **while** $(h \geq 1)$ **do**
13: for all tasks $t_i \in l_h$, compute $VM(t_i)$ using equation 5.11
$mincost[k, t_i] \leftarrow VM(t_i)$;
14: $h \leftarrow h - 1$
15: **end while**
16: $k \leftarrow k + 1$
17: **endwhile**
18: for each assignment, compute $FT(t_{exit})$ using equations 5.2, 5.3 and 5.4 ;
19: select the Pareto solutions among L solutions ;

mesure, dans ces ensembles. L'objectif de cette étape est de définir les différents niveaux du graphe en question [22]. Par conséquent, les tâches appartenant à un même niveau peuvent être exécutées de manière concurrente. Ceci est dû au fait que deux tâches d'un même

22. Nous appelons un niveau d'un graphe un ensemble de tâches sans contraintes de précédence (ne sont pas liées) où tous les successeurs des tâches qui le constitue appartiennent aux niveaux inférieurs

niveau n'échangent pas de données (ou elles ne sont pas liées par des contraintes de précédence). Soit L le nombre de niveaux de l'application (du processus) en question. Le premier niveau l_1 et le dernier niveau l_L contiennent respectivement les tâches t_{input} et t_{output}. Le niveau k, noté l_k, contient toutes les tâches t_j tel que pour chaque arc (t_i, t_j) la tâche t_i appartient à un niveau $k' < k$ et il existe au moins un arc (t_i, t_j) tel que $t_i \in k-1$.

Par exemple, pour le graphe de tâches de la Figure 5.1 il y a cinq niveaux ($L = 5$) tels que :
$l_1 = \{t_1\}$;
$l_2 = \{t_2, t_3, t_4\}$;
$l_3 = \{t_5, t_6\}$;
$l_4 = \{t_7, t_8, t_9\}$;
et $l_5 = \{t_{10}\}$.
La ligne 2 de l'Algorithme 2 correspond à cette étape, à savoir la définition des niveaux du graphe en regroupant des tâches pouvant être exécutées de manière parallèle.

2. Phase d'allocation de ressources

A l'issue de la précédente phase les différentes tâches sont affectées à des ensembles distincts. Les tâches appartenant à un même ensemble (niveau) peuvent être ainsi exécutées de manière concurrente. La deuxième étape de l'approche que l'on propose consiste à choisir une machine virtuelle « optimale » pour exécuter chacune des tâches constituant le processus en question. Ce choix est basé, comme indiqué précédemment, sur les coûts d'exécution et de communication. Étant donné que les tâches appartenant à un même niveau n'échangent pas de données entre elles, l'idée est donc de proposer une heuristique d'affectation de tâches indépendantes. Cependant, et en fonction des tâches qui auraient déjà été affectées, plusieurs stratégies d'exploration du graphe peuvent être distinguées. En effet, le processus d'exploration du graphe consiste à commencer par l'affectation des tâches appartenant à un niveau donné et de continuer à affecter les autres tâches par niveau jusqu'à ce que toutes les tâches du processus soient toutes exécutées. Plus précisément, soit l_k l'ensemble des tâches qui sont affectées en premier. Le niveau du graphe par lequel on commence à explorer le graphe est déterminé par l'une des trois stratégies suivantes de parcours du graphe :

- (a) la stratégie de parcours de haut en bas (*top-down*) ,
- (b) la stratégie de parcours de bas en haut (*bottom-up*),
- et (c) la stratégie de parcours mixte (*mixed strategy*).

(a) **Stratégie de parcours de haut en bas**

Cette stratégie consiste à commencer par choisir la meilleure machine virtuelle pour exécuter la tâche initiale (c.-à-d. t_{input}) en se basant uniquement sur les coûts d'exécution. Ainsi, la machine virtuelle qui exécute la tâche t_{input} est donnée par : $VM(t_{input}) = VM_s$ tel que :

$$EC(t_{input}, VM_s) = \min_{VM_j} EC(t_{input}, VM_j) \tag{5.8}$$

Après cette affectation, le graphe est parcouru de haut en bas, c'est-à-dire en partant du niveau 2 au niveau L. Une tâche qui appartient à un niveau $k \in \{2, .., L\}$ est exécutée par la virtuelle machine VM_s tel que :

$\forall t_i \in l_k, VM(t_i) = VM_s$ tel que :
$EC(t_i, VM_s) + \sum_{t_h \in pred(t_i)} TC(VM(t_h), VM(t_i)) =$

$$\min_{VM_j} \left\{ EC(t_i, VM_j) + \sum_{t_h \in pred(t_i)} TC(VM(t_h), VM(t_i)) \right\} \tag{5.9}$$

où $pred(t_i)$ est l'ensemble des prédécesseurs immédiats de la tâche t_i.

Les lignes 3 (avec $k = 1$) à 10 de l' Algorithme 2 correspondent à cette stratégie de parcours de haut en bas.

(b) **Stratégie de parcours de bas en haut**

Pour cette stratégie, comme son nom l'indique, le graphe est parcouru de bas en haut. Ainsi, nous commençons par le choix de la machine virtuelle minimisant le coût d'exécution de la tâche finale (c.-à-d. t_{output}). La virtuelle machine $VM(t_{input})$ est donnée par :

$VM(t_{output}) = VM_s$ tel que :

$$EC(t_{output}, VM_s) = \min_{VM_j} EC(t_{output}, VM_j) \tag{5.10}$$

Après cette affectation, le graphe est parcouru de bas en haut en affectant les tâches appartenant au niveau $L - 1$ jusqu'au niveau 1. Pour un niveau k, ses tâches sont assignées aux machines virtuelles minimisant les coûts d'exécution et de communication. La virtuelle machine qui exécute une tâche t_i d'un niveau k est donnée par l'équation suivante :

$\forall t_i \in l_k, VM(t_i) = VM_s$ tel que :

$EC(t_i, VM_s) + \sum_{t_h \in succ(t_i)} TC(VM(t_i), VM(t_h)) =$

$$\min_{VM_j} \left\{ EC(t_i, VM_j) + \sum_{t_h \in succ(t_i)} TC\left(VM(t_i), VM(t_h)\right) \right\} \tag{5.11}$$

où $succ(t_i)$ est l'ensemble des successeurs immédiats de la tâche t_i.
Les lignes 11 (avec $k = L$) à 15 de l' Algorithme 2 correspondent a cette stratégie de parcours de bas en haut.

(c) **Stratégie de parcours mixte**

La stratégie mixte, comme son nom l'indique utilise les deux stratégies précédentes simultanément, en commençant par l'affectation des tâches appartenant aux niveaux intermédiaires $k \in \{2, ..., L-1\}$. Plus précisément, soit l_k l'ensemble des tâches par lesquelles on commence le processus d'affectation des ressources. Cette affectation est basée uniquement sur le coût d'exécution. Pour une tâche $t_i \in l_k$, son affectation est donnée par l'équation suivante :
$\forall t_i \in l_k, VM(t_i) = VM_k$ tel que :

$$EC(t_i, VM_k) = \min_{VM_j} EC(t_i, VM_j) \tag{5.12}$$

L'affectation des tâches appartenant aux niveaux $k' < k$ et $k' > k$ est décidée en fonction des affectations précédentes. Ceci dit, les coûts de communication sont aussi pris en compte en plus de ceux d'exécution. Les affectations des tâches appartenant aux niveaux $k_1 < k$ et $k_2 > k$ sont déterminées en utilisant respectivement les stratégies de parcours de haut en bas et de bas en haut. Les Equations 5.9 et 5.11 sont ainsi appliquées de manière récurrente jusqu'à ce que toutes les tâches du processus en question soient affectées.
Les lignes 4 à 17 ($k \in \{2, ..., L-1\}$) de l' Algorithme 2 correspondent à la stratégie de parcours mixte.

Pour récapituler, lorsque la valeur de k vaut 1 ou L (rappelons que L est le nombre de niveaux du graphe de tâches à exécuter) les stratégies de parcours de haut en bas et de bas en haut sont respectivement appliquées. Dans le cas contraire, c'est la stratégie de parcours mixte qui est appliquée.

Notons que l'ensemble des solutions ainsi obtenues est égal à L. En effet, une solution est obtenue en appliquant l'une des premières stratégies. Et la stratégie de parcours mixte permet d'obtenir $L - 2$ solutions.

3. Phase de sélection des solutions non dominées

L'application des stratégies présentées précédemment nous permet d'obtenir L solutions. Pour chacune des solutions nous calculons son temps correspondant en utilisant les équations 5.2, 5.3 et 5.4.
Parmi les couples de solutions nous retenons uniquement les solutions de Pareto dites solutions non dominées sur les deux critères temps et coût.
Les lignes 18 et 19 de l'algorithme correspondent à la phase de sélection des solutions de Pareto

5.4.2 Approche dirigée par la fonction du temps global d'exécution *(global execution time driven-approach)*

Tandis que l'approche précédente est basée sur la fonction objectif du coût global d'exécution, l'approche orienté temps comme son nom l'indique est basée sur la minimisation de la fonction objectif du temps d'exécution et de communication engendrés par l'affectation des différentes tâches aux ressources disponibles. Le nombre de machines virtuelles considéré dans cette approche est infini (non borné). L'approche que l'on propose est composée de trois phases : (i) phase de regroupement des tâches (ii) phase d'allocation et (iii) phase de sélection des solutions efficaces (Pareto).

L'Algorithme 3 donne un aperçu de cette approche.

1. Phase de regroupement des tâches

Cette étape de l'approche que l'on propose est la même que pour l'approche précédente. Rappelons que l'objectif ici est le regroupement des tâches indépendantes (pas de contraintes de précédence et donc pas d'échanges de données).

2. Phase d'allocation de ressources

Cette approche diffère de l'approche précédente dans cette étape. En effet, tandis que l'approche basée sur la fonction objectif du coût global d'exécution optimise le coût de l'utilisation des ressources d'un ensemble de ressources pour exécuter les tâches constituant le workflow en question, l'approche basée sur le temps optimise le temps d'exécution en minimisant le temps global engendré par l'allocation des ressources disponibles.

Les stratégies de parcours décrites dans l'approche basée sur la fonction objectif du coût global d'exécution à savoir : stratégie de parcours de haut en bas, stratégie de par-

Algorithm 3 Approche basée sur le temps global d'exécution

1: read the DAG, the RG and associated attributes values ;
2: sort tasks at each level by traversing the DAG in a top-down fashion ; // let L be the set of levels and l_k the tasks // belonging to the level k
3: $k \leftarrow 1$; // first level
4: **while** $(k \leq L)$ **do**
5: for all tasks $t_i \in l_k$, compute $VM(t_i)$ using equation 5.13
// assign task t_k to the virtual machine $VM(t_i)$
//that minimizes the execution time
$mintime[k, t_i] \leftarrow VM(t_i)$; // *mincost* is a $L \times m$ matrix
// where line k corresponds to the assignment of all tasks
// obtained starting by the tasks assignment belonging to
// this level
6: $h \leftarrow k + 1$; // compute $VM(t_i)$ for all tasks that belong // to levels $h > k$
7: **while** $(h \leq L)$ **do**
8: for all tasks $t_i \in l_h$, compute $VM(t_i)$ using equation 5.14
$mintime[k, t_i] \leftarrow VM(t_i)$;
9: $h \leftarrow h + 1$
10: **end while**
11: $h \leftarrow k - 1$; // compute $VM(t_i)$ for all tasks that belong // to levels $h < k$
12: **while** $(h \geq 1)$ **do**
13: for all tasks $t_i \in l_h$, compute $VM(t_i)$ using equation 5.15
$mintime[k, t_i] \leftarrow VM(t_i)$;
14: $h \leftarrow h - 1$
15: **end while**
16: $k \leftarrow k + 1$
17: **endwhile**
18: for each assignment, compute *cost* using equation 5.7 ;
19: select the Pareto solutions among L solutions ;

cours de bas en haut et stratégie de parcours mixte sont également utilisées pour parcourir le graphe de tâches à exécuter de différentes manières. Nous obtenons ainsi L solutions.

Les ligne 3 (avec $k = 1$) à 10 de l'Algorithme 3 correspondent à la stratégie de parcours de bas en haut. Les lignes 11 (avec $k = L$) à 15 correspondent à la stratégie de parcours de bas en haut. Les lignes 4-17 ($k \in \{2, ..., L - 1\}$) correspondent à la stratégie de parcours

mixte.

Par conséquent, si on commence l'affectation des ressources en exécutant en premier les tâches appartenant au niveau l_k, alors cette affectation est basée uniquement sur le temps d'exécution. Elle est donnée par l'équation suivante :

$\forall t_i \in l, VM(t_i) = r_k$ tel que :

$$ET(t_i, r_k) = \min_{r_j} ET(t_i, r_j) \tag{5.13}$$

Pour les tâches appartenant aux niveaux l_{k+1} et l_{k-1}, les tâches sont affectées aux ressources disponibles en appliquant récursivement les deux équations suivantes :

$\forall t_i \in l_{k+1}, VM(t_i) = r_k$ tel que :

$ET(t_i, r_k) + \sum_{t_h \in pred(t_i)} TT(VM(t_h), VM(t_i)) =$

$$\min_{r_j} \left\{ ET(t_i, r_j) + \sum_{t_h \in pred(t_i)} TT(VM(t_h), VM(t_i)) \right\} \tag{5.14}$$

$\forall t_i \in l_{k-1}, VM(t_i) = r_k$ tel que :

$ET(t_i, r_k) + \sum_{t_h \in succ(t_i)} TT(VM(t_i), VM(t_h)) =$

$$\min_{r_j} \left\{ ET(t_i, r_j) + \sum_{t_h \in succ(t_i)} TT\left(VM(t_i), VM(t_h)\right) \right\} \tag{5.15}$$

Ces deux dernières équations sont appliquées jusqu'à ce que toutes les tâches soient exécutées.

3. Phase de sélection des solutions de Pareto

Rappelons qu'à l'issue de la précédente étape, L affectations des tâches sont obtenues. Dans cette phase, nous calculons d'abord le coût global d'exécution en utilisant l'Equation 5.7) correspondant à chaque solution obtenue et puis uniquement les solutions de Pareto seront maintenues.

5.4.3 Approche basée sur les fonctions du temps et du coût globaux d'exécution (*global execution cost and time driven-approach*)

L'objectif de cette approche est d'aider un utilisateur à choisir les ressources à utiliser pour exécuter un processus scientifique en tenant compte des deux fonctions objectifs (le temps et le coût) simultanément. Le schéma général de cette approche est donné par l'Algorithme 4. Elle est composée de deux étapes :

Algorithm 4 Approche basée sur le temps et le coût

1: Lires le DAG, le RG et les attributs associés ;
2: Exécuter les deux algorithmes Algorithme 2 et 3
3: Sélectionner les solutions dites non-dominées (Pareto)

1. La première étape consiste à exécuter l'Algorithme 2 et l'Algorithme 3 simultanément.
2. La deuxième étape consiste à filtrer les solutions et à ne retenir que celles qui ne sont pas dominées (Pareto).

5.5 Bornes inférieures des fonctions objectifs temps et coût globaux d'exécution

Étant donné que le problème d'allocation de ressources et d'ordonnancement de tâche d'un processus scientifique appartient à la classe NP-complet et afin de mesurer la qualité des solutions que l'on obtient, en appliquant les approches proposées précédemment, nous avons proposé deux bornes inférieures des deux critères pris en compte, à savoir le temps et le coût d'exécution. Pour évaluer la qualité des solutions obtenues par nos algorithmes par les expérimentations présentées ci-après, nous avons besoin de connaître la valeur de la solution optimale pour chaque critère. L'apparence du problème traité à la classe NP-complet, ne nous permet pas d'obtenir les solutions optimales en un temps polynomial. Afin de pallier à ce problème, nous avons défini deux bornes inférieures pour respectivement le temps et le coût globaux engendrés par l'utilisation d'un ensemble de ressources. Autrement dit, au lieu de comparer les solutions obtenues par nos approches avec les solutions optimales, nous les comparons avec des bornes inférieures. Pour des

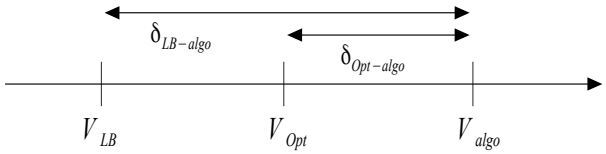

FIGURE 5.4 – Écarts entre les solutions obtenues, les bornes inférieures et les solutions optimales

raisons d'illustration, la Figure 5.4 montre le lien qu'il y a entre la solution optimale, la borne inférieure et la solution obtenue par nos approches. Il est important de rappeler qu'étant donné que les approches proposées sont des heuristiques, les solutions obtenues sont par conséquent des bornes supérieures du problème traité. $\delta_{LB-algo}$ est l'écart entre la solution obtenue en utilisant nos approches (V_{algo}) et la borne inférieure (V_{LB}) et $\delta_{Opt-algo}$ est l'écart effectif entre la solution V_{algo} et la solution optimale.

1. **Borne inférieure de la fonction objectif temps global d'exécution**
   La borne inférieure LB_t est obtenue par l'exécution de la tâche t_{exit} sur le chemin critique du problème relaxé [23] (allocation de ressources et ordonnancement du graphe $G = (E, t)$) défini comme suit :
   — Chaque tâche t_j est exécutée par la machine virtuelle $VM^*(t_j)$ ayant un temps minimum d'exécution :

$$ET(t_j, VM^*(t_j)) = \min_{VM_k} ET(t_j, VM_k) \tag{5.16}$$

   — Chaque arc (t_i, t_j) du graphe G est valué par la borne inférieure du temps de transfert entre les tâches t_i et t_j. Le temps de transfert est donné par l'équation suivante :

$$TT(t_i, t_j) = \min \begin{cases} ET(t_j, VM^*(t_i)) - ET(t_j, VM^*(t_j)) & (1) \\ ET(t_i, VM^*(t_j)) - ET(t_i, VM^*(t_i)) & (2) \\ ET(t_i, t_j, VM_k) & (3) \end{cases}$$

   Avec :

$$ET(t_i, t_j, VM_k) = \min_{VM}\{\alpha + \beta\} \tag{5.17}$$

   Avec,

23. Rappelons que la solution du problème relaxé est une borne inférieure du problème original et qu'elle n'est pas nécessairement par conséquent une solution réalisable

$$\alpha = ET(t_i, VM_l(t_j)) - ET(t_i, VM^*(t_i)) \tag{5.18}$$

$$\beta = ET(t_j, VM_l(t_j)) - ET(t_j, VM^*(t_j)) \tag{5.19}$$

Dans la formule permettant le calcul du temps de transfert $TT(t_i, t_j)$:

— Le premier terme (1) représente le temps d'exécution additionnel de la tâche t_i lorsqu'elle est exécutée sur la machine virtuelle $VM^*(t_j)$ au lieu que ce soit sur la machine virtuelle $VM^*(t_i)$.

— Le deuxième terme (2) correspond au temps d'exécution additionnel de la tâche t_j lorsqu'elle est exécutée sur la machine virtuelle $VM^*(t_i)$ au lieu que ce soit sur sur la machine $VM^*(t_j)$.

— Le troisième terme (3) correspond au temps d'exécution additionnel des tâches t_i et t_j lorsqu'elles sont exécutées sur la machine virtuelle $VM_k(t_i)$ au lieu de $VM^*(t_i)$ et sur la machine virtuelle $VM_k(t_j)$ au lieu de la machine $VM^*(t_j)$ respectivement.

2. **Borne inférieure de la fonction objectif coût global d'exécution**

En ce qui concerne la borne inférieure du coût global engendré par l'utilisation d'un ensemble de ressources, nous utilisons le même principe que pour définir la borne inférieure du temps global d'exécution. Autrement dit, nous utilisons le problème relaxé défini ci-après en remplaçant le temps d'exécution et le temps de transfert par le coût d'exécution et le coût de transfert respectivement. Ainsi, la borne inférieure notée LB_c du coût global correspond à la valeur du coût global du problème relaxé dans lequel :

— Chaque tâche t_j est exécutée sur la machine virtuelle $VM^*(t_j)$ ayant un coût minimum d'exécution :

$$ET(t_j, VM^*(t_j)) = \min_{VM_k} EC(t_j, VM_k) \tag{5.20}$$

— Chaque arc (t_i, t_j) du graphe G est valué par la borne inférieure du coût de transfert entre les tâches t_i et t_j. Le coût de transfert est donné par l'équation suivante :

$$TC(t_i, t_j) = \min \begin{cases} EC(t_j, VM^*(t_i)) - EC(t_j, VM^*(t_j)) & (1) \\ EC(t_i, VM^*(t_j)) - EC(t_i, VM^*(t_i)) & (2) \\ EC(t_i, t_j, VM_k) & (3) \end{cases}$$

avec :

$$EC(t_i, t_j, r_k) = \min_{r_l} \{EC(t_i, r_l(t_i)) - EC(t_i, VM(t_i)) + EC(t_j, r_k(t_j)) - EC(t_j, VM(t_j))\} \tag{5.21}$$

Dans la formule permettant de calculer le coût de transfert $TC(t_i, t_j)$:

— Le premier terme (1) représente le coût d'exécution additionnel de la tâche t_i lorsqu'elle est exécutée sur la machine virtuelle $VM^*(t_j)$ au lieu que ce soit sur la machine $VM^*(t_i)$.

— Le deuxième terme (2) représente le coût d'exécution additionnel de la tâche t_j lorsqu'elle est exécutée sur la machine virtuelle $VM^*(t_i)$ au que ce soit sur la machine $VM^*(t_j)$.

— Le troisième terme (3) correspond au coût de transfert additionnel lorsque les tâches t_i et t_j sont exécutées sur les machines virtuelles $VM_k(t_i)$ et $VM^*(t_i)$ au lieu que ce soit sur les machines virtuelles $VM_k(t_j)$ et $VM^*(t_j)$ respectivement.

5.6 Résultats numériques et discussions

Nous donnons dans cette section un récapitulatif des résultats numériques obtenus en utilisant les approches que l'on propose pour l'allocation de ressources et d'ordonnancement de processus scientifiques dans un environnement du *Cloud Computing*. Nous commençons d'abord par décrire les données de simulation que nous avons utilisées pour générer les différents paramètres du modèle proposé précédemment et discuter ensuite de la qualité des solutions obtenues par nos approches.

5.6.1 Données utilisées pour la simulation

Pour notre expérimentation, nous simulons un environnement du *Cloud Computing* en considérant cinq familles d'instances d'un processus scientifique. Chaque famille est

associée au nombre de tâches qui la composent, c'est-à-dire $n \in \{50, 100, 300, 600, 1000\}$. Chaque famille d'instance est divisée en trois catégories en fonction de la densité de ses contraintes de précédence. Nous appelons ses familles *Small*, *Medium* et *Large*, notées respectivement S, M et L. Ainsi, pour chaque paire de tâches (t_i, t_j) une contrainte de précédence les lie avec une certaine probabilité notée p tel que le graphe obtenu soit sans cycle (DAG). Les probabilités correspondantes pour les séries S, M et L sont respectivement égales à $p_S = 0.2$, $p_M = 0.4$ et $p_L = 0.6$. Le nombre de ressources noté m (machines virtuelle) est défini en fonction du nombre de tâches et sans perte de généralité fixé à $m = n/2$. Nous supposons qu'une tâche du processus en question peut être exécutée par toutes les ressources disponibles. Autrement dit, une tâche t_i du processus considéré peut être affectée et exécutée sur une des catégories des machines virtuelles mises à la disposition de l'utilisateur. Pour toutes les familles d'instances définies précédemment, le temps d'exécution est généré de manière aléatoire et uniformément distribué. Autrement dit, le temps d'exécution des tâches sur les ressources suit une loi de probabilité uniforme dont les valeurs varient entre un et dix. Le coût d'exécution des tâches sur les machines virtuelles suit aussi une loi uniforme définie sur l'intervalle $[0.1, 0.9]$. La bande passante entre les machines virtuelles suit une loi de probabilité uniforme définie sur l'intervalle $[1, 9]$ et la matrice des transferts de données est définie sur l'intervalle $[10, 90]$. Finalement, le coût d'entrée et de sortie des machines virtuelles utilisées suit une loi uniforme définie sur l'intervalle $[1, 9]$. Chaque série de simulation consiste à générer 1000 instances de chaque famille.

5.6.2 Récapitulatif des résultats numériques et discussions

Nous donnons dans ce qui suit un récapitulatif des résultats numériques obtenues en utilisant nos approches. Nous commençons par comparer le ratio des solutions non dominées générées par les trois approches proposées. Nous appelons *Ratio* le nombre de solutions de Pareto, noté $NbreSols$, divisé par le nombre total de solutions noté $NbreTotal$:

$$Ratio = \frac{NbreSols}{NbreTotal}$$

Ce qui détermine le comportement quantitatif des approches proposées, une fois la structure du graphe définie, est le nombre de tâches qui le composent. Nous faisons alors varier le nombre de tâches en considérant les trois types de familles de graphes (Small, Medium et Large). Un récapitulatif des ratios obtenus pour ces trois familles de graphes

est donné par les Figure 5.5, 5.6 et 5.10. Ces résultats nous permettent de conclure deux choses :

1. Le ratio est croissant, dans la plupart des cas, par rapport à la densité des graphes (probabilité p). Ceci est dû à l'importance des échanges de données et par conséquent des temps et des coûts de communication.
2. Le ratio obtenu en utilisant l'approche basée sur le temps et le coût est approximativement égal à la somme des ratios obtenus par les deux approches basées respectivement sur le temps et le coût divisé par deux. Ceci dit aucun des deux critères pris en compte ne domine l'autre. Par conséquent, l'optimisation bi-objectif [170][171][172][97] est la l'approche la plus appropriée pour traiter du problème d'allocation de ressources et d'ordonnancement de tâches des workflows scientifiques dans le cadre du *Cloud*.

Dans ce qui suit, nous discutons de la qualité des solutions obtenues en utilisant nos approches. Les critères de performances que nous utilisons sont définis comme suit :

— D_{min} est obtenu en divisant la valeur minimale, obtenue par une des approches en question, par la borne inférieure.
— $D_{average}$ est obtenu en divisant la moyenne des valeurs, obtenues par une des approches en question, par la borne inférieure.
— D_{max} est obtenu en divisant la valeur maximale, obtenue par une des approches en question, par la borne inférieure.

Les résultats numériques montrent qu'en moyenne la valeur des critères de performances définis ci-dessus n'excédent pas 2.9. Autrement dit, les solutions de Pareto obtenus par nos approches ne dépassent pas trois fois la valeur de la borne inférieure.

5.7 Conclusion

Dans ce chapitre, nous avons proposé trois approches complémentaires pour l'allocation de ressources et l'ordonnancement des tâches d'applications dont toutes les tâches sont supposées automatisées. C'est le cas en effet lorsque l'on considère les processus scientifiques (workflows scientifiques). Autrement dit, l'ensemble des ressources sont des machines virtuelles. Dans les approches proposées, nous avons pris en compte en plus des temps et des coûts d'exécution, les temps et les coûts de transfert. La première approche est guidée par la fonction objectif temps, la deuxième est basée sur la fonction objectif coût engendré par l'utilisation d'un ensemble de ressources. La troisième approche est

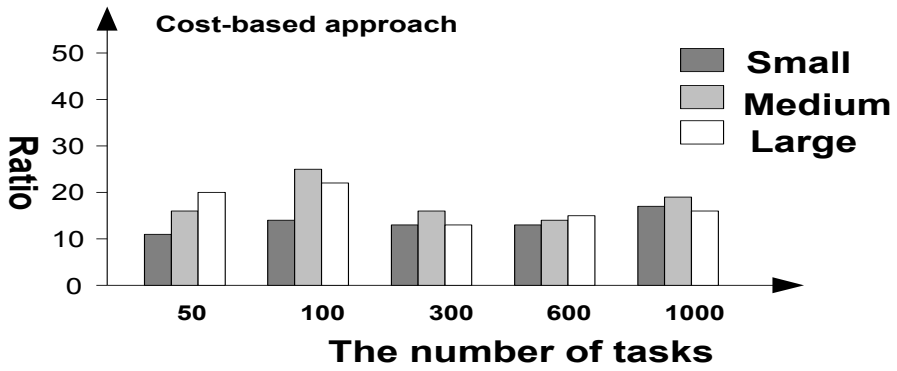

FIGURE 5.5 – Le ratio des solutions de Pareto obtenues avec l'approche basée sur le coût en fonction du nombre de tâches des workflows (100 processus générés de manière aléatoire)

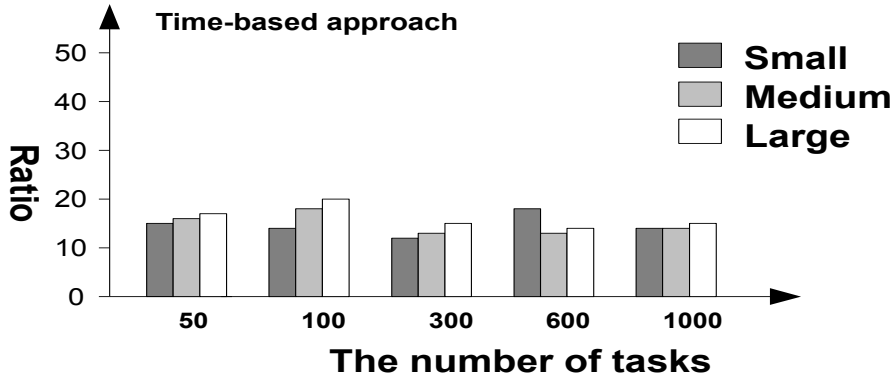

FIGURE 5.6 – Le ratio des solutions de Pareto obtenues avec l'approche basée sur le temps en fonction du nombre de tâches des workflows (100 processus générés de manière aléatoire)

une combinaison des deux premières approches dans le sens où les deux fonctions objectifs sont prises en compte simultanément et dont l'objectif est la sélection uniquement des solutions dites non-dominées (Pareto). De plus, dans le but d'évaluer la qualité des solutions obtenues par nos approches et étant donné l'appartenance du problème traité à la classe NP-complet [167][168][169], nous avons proposé deux bornes inférieures pour les deux fonctions objectifs prises en compte dans nos travaux. En conclusion, contrairement aux approches existantes pour l'ordonnancement des graphes de tâches, nous avons pris en compte deux critères de qualité de service conflictuels en plus de l'une des caractéristiques les plus importantes du *Cloud* à savoir l'« illusion de ressources infinies ».

Nous étudions dans le chapitre suivant le cas où toutes les tâches ne sont pas automa-

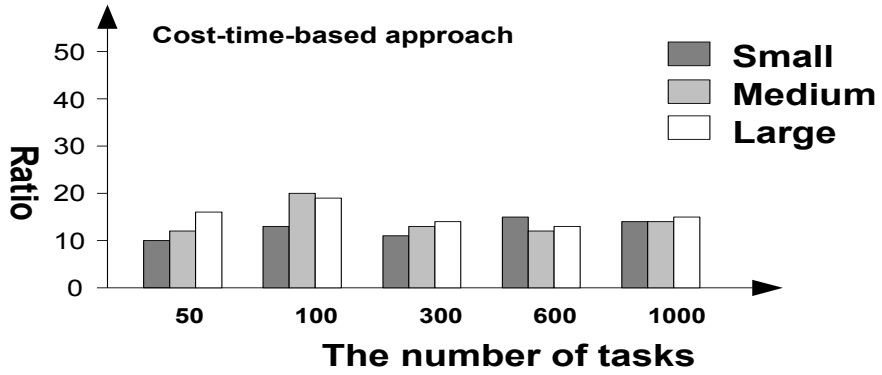

FIGURE 5.7 – Le ratio des solutions de Pareto obtenues avec l'approche basée sur les fonctions objectifs temps et coût en fonction du nombre de tâches des workflows (100 processus générés de manière aléatoire)

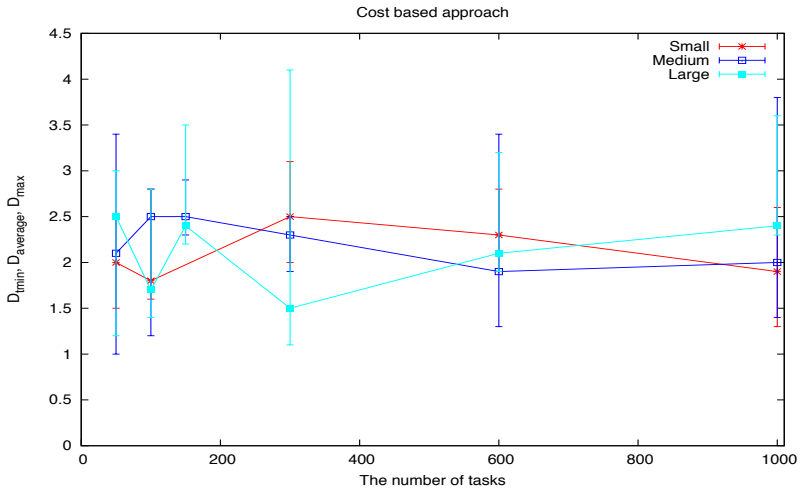

FIGURE 5.8 – Le ratio des solutions de Pareto obtenues avec l'approche basée sur les fonctions objectifs temps et coût en fonction du nombre de tâches des workflows (100 processus générés de manière aléatoire)

tisées en considérant deux types de ressources : (1) machines virtuelles et (2) ressources humaines. L'introduction de cette dernière engendre le problème de la gestion de l'« opacité » de ces dernières que nous proposons de capturer en utilisant des modèles de prévision pour l'estimation de leurs disponibilités.

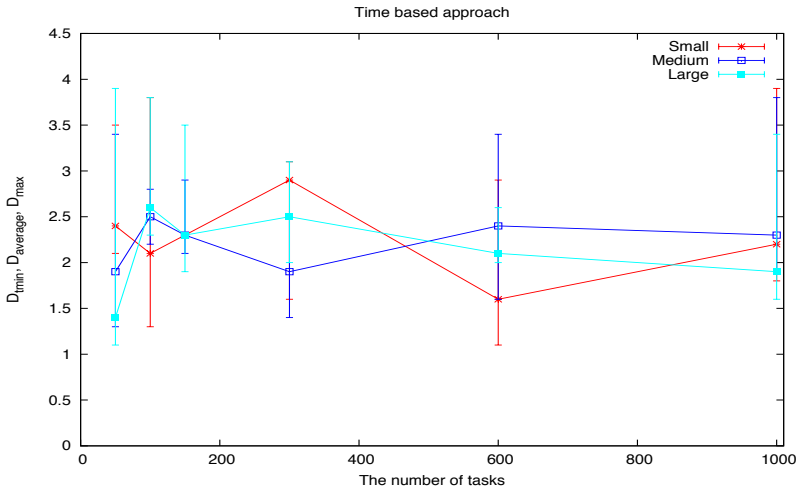

FIGURE 5.9 – Le ratio des solutions de Pareto obtenues avec l'approche basée sur les fonctions objectifs temps et coût en fonction du nombre de tâches des workflows (100 processus générés de manière aléatoire)

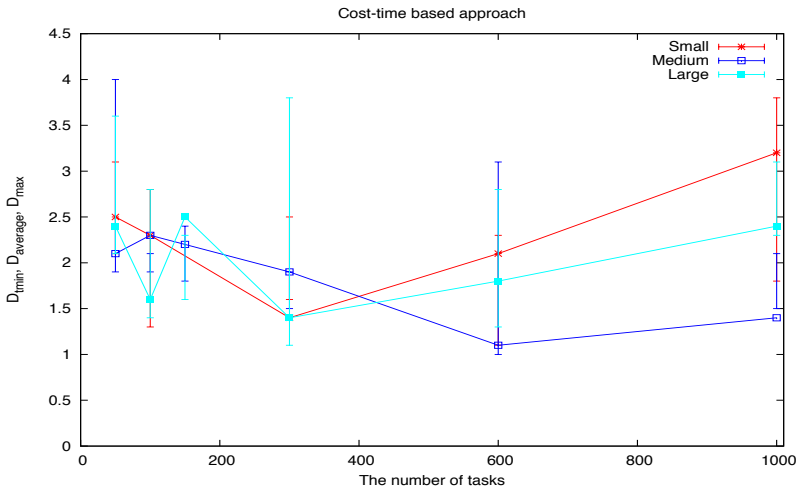

FIGURE 5.10 – Le ratio des solutions de Pareto obtenues avec l'approche basée sur les fonctions objectifs temps et coût en fonction du nombre de tâches des workflows (100 processus générés de manière aléatoire)

Chapitre 6

Optimisation de l'exécution d'une instance d'un modèle de processus métier

Sommaire

6.1 Introduction

Nous distinguons généralement deux types de processus, à savoir (i) les processus scientifiques et (ii) les processus métiers. Tandis que le chapitre précédent est consacré à l'allocation de ressources et l'ordonnancement de tâches d'un processus scientifique, nous nous intéressons dans celui-ci à la même problématique lorsque l'on exécute un processus métier. Les différences principales, en ce qui concerne la gestion des ressources en général et le problème d'ordonnancement en particulier, entre ces deux types de processus sont résumés par les points suivants :

— Généralement toutes les tâches constituant un processus scientifique sont automatisées tandis qu'un sous ensemble de tâches d'un processus métier nécessite l'intervention de ressources humaines. En effet, à cause de la nature des applications métiers il est rare, voire impossible, d'avoir un modèle de processus métier où toutes les tâches qui le composent sont automatisées.

— L'introduction de ressources humaines (en nombre limité) engendre le problème d'ordonnancement des tâches au niveau de chaque file d'attente des ressources utilisées. D'autre part, ces ressources peuvent être sollicitées pour exécuter d'autres tâches dont on n'a pas le contrôle. Autrement dit, une ressources donnée peut être amenée à exécuter des tâches provenant d'autres processus autre que celui qui nous intéresse.

— Lors de l'exécution d'un processus scientifique, généralement une seule instance est lancée alors qu'il est rare de n'exécuter qu'une seule instance d'un processus métier. En effet, généralement plusieurs instances sont exécutées simultanément.

Nous nous focalisons dans ce qui suit sur les deux premiers points. Le troisième point est l'objet du chapitre suivant. Plus précisément, nous nous intéressons dans ce qui suit au problème d'allocation de ressources et d'ordonnancement de tâches d'une instance d'un processus métier en distinguant deux types de tâches, à savoir les tâches automatisées (pouvant donc être exécutée par des machines virtuelles) et les tâches non automatisées (nécessitant donc l'intervention de ressources humaines pour les accomplir). De plus, le modèle que l'on propose permet de prendre en compte le fait qu'une ressource peut être amenée à exécuter d'autres tâches. Pour ce dernier point, nous proposons d'utiliser des modèles de prévision afin de prédire la disponibilité d'une ressource donnée.

Pour récapituler, les apports de ce chapitre sont résumés par les points suivants :

1. Nous distinguons deux types de tâches :

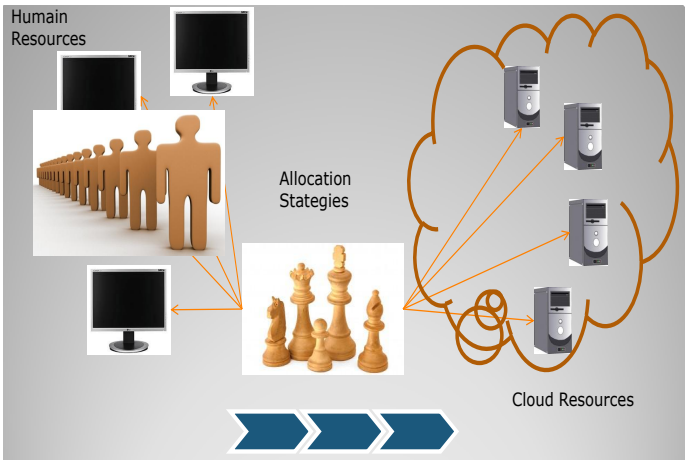

FIGURE 6.1 – Vue globale du problème traité dans ce chapitre

(a) Tâches automatisée : ces tâches sont exécutées par des machines virtuelles. Etant donnée l'une des caractéristiques principales du *Cloud computing*, à savoir « illusion de ressources infinies », l'affectation des tâches au niveau des machines virtuelles ne se pose pas. En effet, à chaque affectation d'une tâche à une machine virtuelle, une instance de cette dernière est créée. En ce qui concerne le nombre de machines virtuelles, mises à la disposition des utilisateurs, elles sont supposées être en nombre infini. Cependant, il convient de souligner que le nombre de types de machines virtuelles est fini.

(b) Tâches non automatisées : une gestion des ressources humaines pour l'exécution de ce type de tâches est nécessaire. Etant donné le nombre fini de ressources humaines disponibles, une gestion optimale de leur file d'attente est nécessaire. Dans ce contexte, ***nous nous intéressons donc à un problème d'allocation de ressources et d'ordonnancement de tâches à ressources limitées***.

2. Prise en compte du fait que les ressources humaines peuvent être sollicitées par d'autres processus et ceci de manière « opaque » pour les systèmes gestion de workflows qui les orchestrent. Autrement dit, un système de gestion de workflow donné n'a pas toutes les informations sur la file d'attente d'une ressource au moment de son utilisation. Afin donc de prendre en compte cet aspect, nous proposons d'utiliser des modèles de prévision pour estimer la disponibilité des ressources humaines avant l'affectation des tâches.

Les approches que l'on propose dans ce chapitre pour optimiser la gestion de l'allocation des ressources et de l'ordonnancement des tâches d'une instance d'un processus métier sont composées de quatre étapes (une vue global du problème traité dans ce chapitre est donnée par la Figure 6.1) :

1. La première étape consiste à construire un graphe de tâches sans cycle à partir de la spécification du processus à exécuter. Notons que dans ce qui suit nous ne nous intéressons qu'à l'exécution d'une seule instance du processus donné. Les ressources disponibles peuvent être éventuellement sollicitées par d'autres instances qui s'exécutent de manière concurrente.
2. Une fois le graphe de tâche construit, l'étape suivante du processus d'allocation de ressources que l'on propose permet de prédire le nombre d'instances à exécuter de manière concurrente et donc de déduire les disponibilités des ressources utilisées. Plus précisément, l'objectif de cette étape est d'utiliser un modèle de prévision afin d'estimer le temps moyen d'exécution d'une tâche donnée par une ressource humaine.
3. Etape d'allocation des ressources. Á l'issue de l'étape précédente et en fonction de la stratégie d'allocation retenue une décision est prise quant à l'affectation d'une tâche sur une ressource donnée.
4. Sélection des solutions dites non-dominées (ou de Pareto).

La suite de ce chapitre est structurée comme suit. Dans la section suivante, nous proposons un modèle pour l'optimisation de l'utilisation des ressources disponibles pour l'exécution d'une instance d'un modèle de processus métier. Nous exposons les différents paramètres pris en compte dans notre modélisation ainsi que les deux fonctions objectifs prises en compte. Étant donné l'opacité de la gestion des ressources notamment humaines liée au fait qu'elles peuvent être sollicitées pour exécuter des tâches de plusieurs processus métiers, nous proposons dans la Section 6.3 une heuristique basée sur le lissage exponentiel afin de mieux gérer les files d'attente des ressources utilisées. La Section 6.5 est consacrée à la présentation des approches d'allocation de ressources et d'ordonnancement des tâches d'une instances d'un modèle de processus métier en distinguant deux types de ressources (machines virtuelles et ressources humaines). La Section 6.6 fait un récapitulatif des résultats numériques obtenus en mettant l'accent sur l'importance d'utiliser une heuristique pour prédire la disponibilité des ressources. Le chapitre est conclu par la Section 6.7.

6.2 Modélisation du problème d'exécution d'une instance d'un modèle de processus métier

Avant d'introduire les différents paramètres du modèle proposé ainsi que les fonctions objectifs, prises en compte dans les approches d'optimisation d'allocation de ressources et d'ordonnancement de tâches d'une instance d'un modèle de processus métier, nous commençons par décrire les principales raisons qui motivent les hypothèses sur lesquelles s'appuient notre démarche. Ensuite, nous donnons les définitions des différents paramètres pris en compte dans le modèle que l'on propose. Contrairement aux workflows scientifiques lorsque l'on exécute un workflow métier il est rare voir impossible d'automatiser toutes ses tâches. Par conséquent, il convient de distinguer deux tâches de deux natures différentes :

1. Des tâches automatisées : leur exécution ne nécessite pas l'intervention de ressources humaines. Dans le cadre qui nous intéresse, à savoir le *Cloud Computing*, ces dernières sont exécutées sur des machines virtuelles.
2. Des tâches non automatisées (ou manuelles) : leur exécution nécessite l'intervention de ressources humaines.

De plus, les ressources humaines disponibles peuvent être sollicitées pour l'exécution d'autres tâches qui n'appartiennent pas à l'instance du modèle de processus qui nous intéresse. Le nombre de tâches susceptibles d'être exécutées par une ressource humaine donnée ne peuvent être déterminé de manière exacte. Par conséquent, l'orchestration par les systèmes de gestion de workflows de ce type de ressources sans avoir des informations sur leur charge de travail peut dégrader de manière considérable les performances des applications qui les utilisent. Par ailleurs, plusieurs interactions entre les ressources et les différents systèmes de gestion de workflows rendent cette tâche difficile du au fait notamment des phénomènes aléatoires qui peuvent surgir lors de l'exécution d'une application donnée (absence, congés, difficulté à réaliser une tâche, ...) [100][102][103][104][105]. Afin d'y remédier, nous proposons une heuristique pour l'estimation de la disponibilité des ressources humaines. Cette heuristique est basée sur le lissage exponentiel afin de prendre en compte les phénomènes aléatoires sous-jacents à la gestion des ressources humaines. Les différents paramètres pris en compte dans notre modélisation sont donnés par les définitions suivantes.

Définition 13 (Ressource) *Une ressource r représente une unité disponible requise pour la réalisation d'une tâche. Comme mentionné précédemment une ressource peut être une*

ressource humaine (HR) ou machine virtuelle (VM). L'ensemble des ressources disponible est noté R

Définition 14 (Rôle) *Un rôle ro se réfère à un ensemble de ressources qui ont les mêmes compétences. L'ensemble des rôles est noté Ro.*

Définition 15 (Tâche) *Une Tâche t_i est une unité logique de travail réalisée par une ressource qui peut être une ressource humaine (dans le cas où la tâche n'est pas automatisé) ou une machine virtuelle (dans le cas où la tâche est automatisée)*

Définition 16 (Processus Métier) *Un processus métier est représenté par un graphe acyclique (DAG). Formellement, c'est un DAG $G = (T, E)$, avec :*

- *$T = \{t_1, ..., t_n\}$ est l'ensemble fini des tâches qui le compose, pouvant être réalisé par une HR ou une VM.*
- *E représente l'ensemble des arcs qui le composent, représentant les contraintes de précedence et/ou de données entre les tâches de l'ensemble T.*
- *Un arc (t_i, t_j) du graphe G existe si et seulement s'il existe une contrainte temporelle et /ou de données entre les tâches t_i et t_j. Si une contrainte temporelle existe entre ces deux tâches alors t_j ne peut être exécutée que lorsque la tâche t_i a été exécutée.*

Définition 17 (Graphe de ressources) *Les ressources disponibles sont modélisées par un graphe RG. Soit un graphe de ressources $RG = (R, V)$, avec*

- *$R = \{VM_1, ..., VM_m, HR_1, ..., HR_{m'}\}$ l'ensemble fini de machines virtuelles types et de ressources humaines.*
- *V représente l'ensemble des arcs entre les différentes catégories de machines virtuelles. Chaque arc (VM_i, VM_j) représente le lien entre les deux machines virtuelles VM_i et VM_j.*

Rappelons également que le nombre de ressources humaines est supposé fini et que nous pouvons avoir autant d'instances de machines virtuelles que nécessaires pour exécuter une application donnée.

Définition 18 (Disponibilité) *Nous appelons disponibilité (*availability*) d'une ressource r_j le temps qui s'est écoulé depuis le commencement de l'exécution de l'instance du modèle de processus en question jusqu'à la date de sa disponibilité pour exécuter une autre tâche t_i. Elle est notée par $Avail[r_j, t_i]$. Il est à noter que la disponibilité d'une ressource de type machine virtuelle est nulle. En effet, ceci est une conséquence directe de l'hypothèse sur le nombre infini de machines virtuelles mises à la disposition des utilisateurs.*

Notation	Définition
$r(t_i)$	la ressource qui exécute la tâche t_i
$TT(r(t_i), r(t_j))$	le temps de transfert entre la ressource $r(t_i)$ et $r(t_j)$
$ET(t_i, r_j)$	le temps d'exécution de la tâche t_i par la ressource r_j
$EC(t_i, r_j)$	le coût d'exécution de la tâche t_i par la ressource r_j
$TC(r(t_i), r(t_j))$	le coût de transfert entre les ressource $r(t_i)$ et $r(t_j)$

TABLE 6.1 – Notations et définitions des différents paramètres du modèle proposé

Les notations utilisées, dans la suite de ce chapitre, ainsi que leurs définitions sont données par le tableau 6.1.

Ainsi, le problème auquel nous nous intéressons dans ce chapitre peut être résumé comme suit. Étant donné un ensemble de ressources composé de machines virtuelles et de ressources humaines avec différents rôles (un ensemble de rôle) et une instance d'un modèle de processus représentée sous forme d'un graphe de tâches sans cycle. L'objectif est de déterminer une affectation des tâches constituant l'instance en question tout en minimisant deux fonctions objectifs conflictuelles, à savoir le temps global d'exécution et le coût global d'exécution. Avant de décrire ces deux fonctions objectifs, nous décrivons ci-après l'heuristique que l'on propose pour prédire la disponibilité des ressources utilisées.

6.3 Heuristique pour l'estimation de la charge de travail des ressources humaines

L'objectif de ce chapitre est de proposer des approches pour l'allocation de ressources et l'ordonnancement de tâches d'une instance d'un modèle de processus donné tout en tenant compte de deux critères de qualité de service (le temps et le coût d'exécution). De plus, et contrairement au chapitre précédent, nous distinguons deux types de ressources (machines virtuelles et ressources humaines). Il est important de rappeler que les ressources humaines peuvent être amenées à exécuter d'autres tâches n'appartenant pas à l'instance à laquelle nous nous intéressons. Par conséquent, le temps que met une ressource r_j pour exécuter une tâche t_i dépend du nombre de tâches que cette ressource doit exécuter avant d'entamer l'exécution de la tâche t_i. Ceci est dû au fait que le nombre de ressources humaines disponibles est en nombre fini.

Dans le souci d'améliorer la qualité des solutions que l'on propose notamment pour le

calcul du temps global d'exécution, nous introduisons une heuristique pour l'estimation de la disponibilité des ressources humaines sollicitées. Cette heuristique est justifiée par le fait qu'une ressource humaine donnée peut être amenée à exécuter d'autres tâches n'appartenant pas au processus auquel nous nous intéressons.

Afin de prendre en compte cet aspect, deux approches peuvent être utilisées [24] :

— Estimer le temps d'exécution de la tâche t_i par la ressource r_j.

— Estimer la disponibilité de la ressource r_j pour l'exécution de la tâche t_i.

Nous retenons dans la suite de ce chapitre la deuxième approche pour illustrer l'heuristique que l'on propose.

Il est indispensable de disposer de méthodes permettant d'estimer la charge de travail de ce type de ressources (i.e. ressources humaines). Nous proposons dans ce qui suit une heuristique basée sur l'observation des charges de travail de chacune des ressources humaines sollicitées afin de prédire la valeur de sa disponibilité future.

Le principe de l'heuristique que l'on propose pour estimer la charge de travail des ressources humaines qui interviennent dans l'exécution d'une instance d'un processus métier est basé sur le lissage exponentiel. Plus précisément, notre objectif est de déterminer la date de la disponibilité d'une ressource humaine r_j pour l'exécution d'une tâche t_i sachant que cette même tâche a été exécutée par r_j N fois dans le passé. Nous notons cette date par $Avail[j,i](k)$ tel que $k > N$. Nous disposons de N observations notées $Avail[j,i](k), 0 \leq k \leq N$. En se basant sur ces observations, notre objectif est de prédire la valeur future de la disponibilité de la ressource r_j pour exécuter la tâche t_i (i.e. $Avail[j,i](k)$). Une idée intuitive serait d'affecter pour cette dernière la valeur moyenne des observations précédentes. Formellement, cette valeur est donnée par :

$$Avail[j,i](k) = \frac{\sum_{l=1}^{l=k-1} Avail[j,i](l)}{k-1}$$

Généralement, l'utilisation de la moyenne pour l'estimation des valeurs futures de la disponibilité conduit à leur sous-estimation. L'idée du lissage exponentiel est de donner un poids d'autant moins important que les observations sont loin dans le passée, avec une décroissance exponentielle. Avant de donner la formule permettant de calculer la valeur future de la disponibilité d'une ressource avant son utilisation, nous commençons par construire une liste dite lissée des observations disponibles. Afin de donner un poids plus important aux observations récentes, nous introduisons un paramètre $0 < \beta < 1$, appelé paramètre de lissage, tel que :

24. Ces deux approches ont un même objectif, à savoir l'estimation de la fonction du temps global d'exécution

— $0 \leq \beta \leq 1$

— β proche de 1, cela signifie que les observations passées sont plus prépondérantes que les observations récentes.

— β proche de 0, cela signifie que les observations récentes ont un poids plus important que les observations loin dans le passé.

La prévision de la disponibilité de la ressource r_j notée par $Avail\hat{[}j,i](k)$ est donnée par l'équation suivante :

$$Avail\hat{[}j,i](k) = (1-\beta)\sum_{l=0}^{k-1}\beta^j Avail[j,i](k-l) \tag{6.1}$$

L'utilisation de la méthode adaptative de mise à jour (ordre 1) donne les résultats suivants :

$$Avail\hat{[}j,i](k) = (1-\beta)Avail[j,i](k) + \beta Avail[j,\hat{i}](k-1) \tag{6.2}$$

En effet, on a d'après l'équation 6.1 :

$$\beta Avail[j,\hat{i}](k-1) = (1-\beta)\sum_{j=0}^{k-2}\beta^{j+1}Avail[j,i](k-(j+1)) \tag{6.3}$$

Un changement de variable tel que j'=j+1 (donc $1 \leq j' \leq k-1$) permet d'obtenir le résultat suivant :

$$\beta Avail[j,\hat{i}](k-1) = (1-\beta)\sum_{j'=1}^{k-1}\beta^{j'}Avail[j',i](k-j') \tag{6.4}$$

Alors,

$$Avail\hat{[}j,i](k) - \beta Avail[j,\hat{i}](k-1) = (1-\beta)\left\{\sum_{j=0}^{k-1}\beta^j Avail[j,i](k-j) - \sum_{j=1}^{k-1}\beta^j Avail[j,i](k-j)\right\} \tag{6.5}$$

D'où l'Equation 6.2.

La formule itérative utilisée pour calculer la série lissée et la prévision sont données par les équations suivantes [25] :

25. Afin d'alléger les notations, on pose $Avail\hat{[}j,i](k) = F_{k-1}$

$$\begin{cases} F_0 = Avail[j,i](1) \;\; ou \;\; X_0 = \sum_{l=1}^{N} Avail[j,i](l)/N \\ \\ F_{t+1} = (1-\beta)X_t + \beta F_t, 0 \leq t \leq N \\ \\ F_t = F_{N+1}, t \geq N+1 \end{cases} \tag{6.6}$$

Le choix de la valeur du paramètre α relève généralement de considérations empiriques. Une des méthodes les plus utilisées est la minimisation des moindres carrées des erreurs (prévision/réalisation). Sinon, il est évident qu'une valeur proche de 1 du paramètre β donne de meilleurs résultats à court terme qu'à long terme.

L'Algorithme 5 présente l'heuristique que l'on propose pour prédire la disponibilité des ressources humaines utilisées dans la mise à jour de la valeur de la fonction objectif du temps global d'exécution. Cette fonction est décrite ci-après. Un exemple numérique sur l'application de cette heuristique pour le calcule de la fonction objectif du temps global d'exécution est donné après la description ci-après de cette dernière.

Algorithm 5 Predictive heuristic for the availability of resources

1: Let a set of previous observations of the availability of a given resource r_j and a task t_i ;
2: Smooth the previous availability using Equation 6.6 ;
3: Predict the value of the resource r_j availability for the task t_i execution using Equation 6.2 ;
4: Use this value to compute the overall execution time ;

6.4 Les fonctions objectifs prises en compte

Nous décrivons dans ce qui suit les deux fonctions objectifs prises en compte. Nous illustrons à travers un exemple numérique l'utilisation de l'heuristique décrite précédemment pour calculer la fonction objectif du temps global d'exécution.

6.4.1 La fonction objectif du temps global d'exécution

L'objectif ici est de proposer des stratégies d'allocation de ressources et d'ordonnancement d'une instance d'un processus métier tout en distinguant deux types de tâches :

— Tâches non automatisées (ou manuelles), exécutée par des ressources humaines (HR).
— Tâches automatisées, exécutées par des machines virtuelles (VM).

Ces ressources, humaines notamment, peuvent être utilisées (sollicitées) pour l'exécution d'autres tâches n'appartenant pas à l'instance du processus à laquelle nous nous intéressons. Par conséquent, pour le calcul (ou plus précisément la mise à jour) de la fonction objectif du temps global d'exécution deux cas sont à distinguer :

A. La tâche courante t_j est une tâche automatisée

Dans ce cas la tâche t_j est exécutée en utilisant une machine virtuelle. Étant donnée l'hypothèse sur le nombre illimité de machines virtuelles mises à la disposition des utilisateurs, pour l'exécution de la tâche courante une autre machine virtuelle est instanciée et commence immédiatement l'exécution de la tâche en question à condition que toutes les tâches qui la précédent soient exécutées et que toutes les données nécessaires à l'exécution de la tâche t_j soient transmises de façon à prendre en compte les temps de transfert.

B. La tâche courante t_j est une tâche manuelle

Dans le cas où une ressource humaine est utilisée pour exécuter la tâche courante et étant donné le nombre limité de ressources humaines disponibles, lorsqu'une de ces ressources est occupée (en train d'exécuter une tâche t_j) une autre tâche ne lui sera affectée que lorsque elle termine d'exécuter la tâche t_j. De plus, toutes les tâches qui précédent la tâche courante (i.e. la tâche t_j) doivent être exécutées (prise en compte des contraintes de précédence).

Étant donné l'hypothèse sur l'infinité de machines virtuelles mises à la disposition des utilisateurs, le problème de gestion optimale de l'opacité de la charge de travail des ressources ne concerne donc pas ces dernières mais uniquement les ressources humaines du fait qu'elles sont en nombre fini. En effet, le *Cloud* permet à un utilisateur donné d'instancier (créer) autant d'images de machines virtuelles que nécessaires pour l'exécution des ses applications.

Avant d'énoncer la fonction objectif du temps global d'exécution, nous introduisons les quatre paramètres suivants. Soit un ordonnancement partiel (c'est-à-dire juste un sous-ensemble de tâches, de l'instance à exécuter, sont accomplies) :

— La date de début au plus tôt de l'exécution d'une tâche t_j notée EST (*earliest start time*).

— La date de fin au plus tard de l'exécution de la tâche t_j notée EFT (*earliest finish time*).

— La disponibilité d'une ressource r_j notée $avail[j]$ (*availability*).

— La date effective de fin d'exécution de la tâche t_j notée AFT (*actual finish time*).

Pour calculer les deux premiers paramètres pour une tâche donnée de l'instance du modèle de processus à exécuter, nous avons besoin de connaître les tâches ayant déjà été exécutées et les ressources utilisées pour les exécuter. Notons que cela dépend de la stratégie d'allocation (voir plus loin) utilisée pour le choix des ressources. De plus, si une tâche donnée est affectée à une ressource humaine, la disponibilité de cette dernière est prédite en utilisant l'heuristique proposée auparavant. Plus précisément, ces quatre paramètres sont calculés comme suit. Pour la première tâche de l'instance du modèle de processus à exécuter (i.e. t_{input}), son temps d'exécution au plus tôt et au plus tard sont donnés par respectivement les deux équations suivantes :

$$EST(t_{input}, r_j) = 0 \tag{6.7}$$

$$EFT(t_{input}, r_j) = ET(t_i, r_j) \tag{6.8}$$

Pour les autres tâches du graphe, les temps d'exécutions au plus tôt et au plus tard sont calculés de manière récursive, en commençant par la tâche initiale, en utilisant les deux équations suivantes :

$$EFT(t_i, r_j) = EST(t_i, r_j) + ET(t_i, r_j) \tag{6.9}$$

$$EST(t_i, r_j) = \max\left\{avail[j], \max_{t_p \in pred(t_j)}[AFT(t_p) + TT(r(t_p), r(t_i))]\right\} \tag{6.10}$$

Avec :

— $pred(t_i)$ l'ensemble des prédécesseurs immédiats de la tâche t_i,

— $avail[j]$ la date où la ressource r_j est disponible (prête à commencer l'exécution d'une autre tâche). Etant donné l'hypothèse sur le nombre de machines virtuelles mise à la disposition des utilisateurs, il est claire que $avail[j] = 0$ pour toutes les machines virtuelles utilisées. En d'autres termes, si la tâche t_i est exécutée par une ressource r_j de type machine virtuelle, alors la date de début au plus tôt $EST(t_i, r_j)$ est définie par l'équation suivante :

$$EST(t_i, r_j) = \max_{t_p \in pred(t_j)} \{AFT(t_p) + TT(r(t_p), r(t_i))\} \tag{6.11}$$

Pour les deux derniers paramètres liés directement aux ressources utilisées (i.e. $AFT(t_j)$ $Avail[j]$), ils ne sont connus qu'une fois les affectations des tâches sont effectives.

Exemple d'illustration

Afin d'illustrer le calcul de ces paramètres, soit une instance d'un processus composé de quatre tâches (t_1, t_2, t_3 et t_4) représentées par la Figure 6.2. Soit un ordonnancement tel que $r(t_1) = r_1$, $r(t_2) = r_2$, $r(t_3) = r_1$ et $r(t_4) = r_2$. Les dates de début au plus tôt, les dates de fin au plus tôt et les dates de fin effectives des ces quatre tâches ainsi que les disponibilités des ressources, r_1, r_2, r_3 et r_4 sont donnés ci-après :

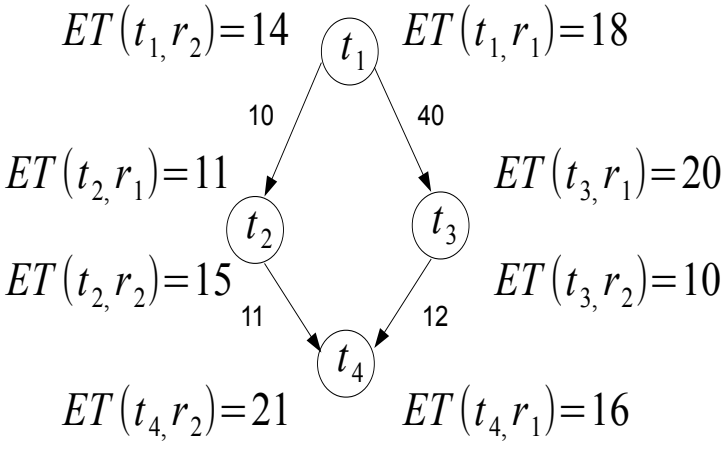

FIGURE 6.2 – Exemple d'une instance d'un processus métier

— Pour la tâche t_1 ces dates de début au plus tôt et de fin au plus tard sont données par $EST(t_1, r_1) = 0$ et $EFT(t_1, r_1) = ET(t_1, r_1) = 14$. Autrement dit, la tâche t_1 peut être exécutée immédiatement par la ressource r_1. Si la tâche t_1 est exécutée par la ressource r_2, alors les dates de début au plus tôt et de fin au plus tard sont $EST(t_1, r_2) = 0$ et $EFT(t_1, r_2) = ET(t_1, r_2) = 18$. Rappelons qu'ici on a supposé que la tâche t_1 est exécutée par la ressource r_1. Par conséquent, la date de fin actuelle de la tâche t_1 est donnée par $AFT(t_1) = 14$. Finalement, la disponibilité de la ressource r_1 est $Avail[1] = 14$ (cette ressource ne peut exécuter une autre tâche avant l'instant $t = 14$). Par contre la disponibilité de la ressource $r_2 = 0$ (cette ressource est prête à commencer l'exécution d'éventuelles tâches arrivant dans sa

file d'attente).

— Quant à la tâche t_2, ses dates de début au plus tôt sont données par $EST(t_2, r_1) = \max\{14, 14 + 10\} = 24$ et $EST(t_2, r_2) = \max\{0, 14 + 10\} = 24$. Ceci signifie que quelque soit la ressource qui exécute la tâche t_2, cette dernière ne peut être exécutée (plus précisément commencée à être exécutée) qu'à partir de l'instant $t = 24$. Ceci vient du fait que cette tâche doit attendre que toutes les tâches qui la précèdent (ici l'ensemble est réduit à une seule tâche, à savoir t_1) terminent de s'exécuter et que toutes les données requises à son exécution soient disponibles. Rappelons que dans le cas où deux tâches sont exécutées sur la même machine virtuelle, alors le temps de transfert est nul. Supposons que t_2 est exécutée par la ressource r_2, alors son temps de fin est $EFT(t_2, r_2) = EST(t_2, r_2) + ET(t_2, r_2) = 24 + 15 = 39$. Par conséquent, $Avail[2] = 39$ et $AFT(t_2) = 39$.

— Pour la tâche t_3, $EST(t_3, r_1) = \max\{14, 14 + 40\} = 54$
et $EST(t_3, r_2) = \max\{39, 14 + 40\} = 54$. La tâche t_3 ne peut donc être exécutée qu'à partir de l'instant $t = 54$. Si la tâche t_3 est exécutée par la ressource r_1, alors $EFT(t_3, r_1) = 54 + 20 = 74$. Par conséquent, $Avail[1] = 74$.

— Pour la dernière tâche (i.e. t_4),
$EST(t_4, r_1) = \max\{74, \max\{39 + 11, 74 + 12\}\} = 86$
et $EST(t_4, r_2) = \max\{39, \{39 + 11, 74 + 12\}\} = 86$. Si toutefois les tâches sont exécutées sur des machines virtuelles qui se trouvent sur le même site, alors le temps de transfert devient nul vu notre hypothèse de départ. La tâche t_4 ne peut donc être exécutée qu'à partir de l'instant $t = 86$. Si la tâche t_4 est exécutée par la ressource r_1, alors $EFT(t_4, r_1) = 86 + 16 = 102$. Par conséquent, $Avail[1] = 102$. Cette dernière valeur correspond au temps global d'exécution de l'instance en question.

Un récapitulatif des valeurs des quatre paramètres introduits précédemment, sans prise en compte des prévisions des disponibilités des ressources, obtenues pour l'ordonnancement de cet exemple est donné par le tableau 6.2.

Il est évident que lorsque ces ressources peuvent être sollicitées pour exécuter d'autres tâches, leurs disponibilités ne sont plus les mêmes. En effet, prenant le cas de la tâche t_1

	$EST(t_j, r_1)$	$EST(t_j, r_2)$	$EFT(t_j, r_1)$	$EFT(t_j, r_2)$	$AFT(t_j)$
t_1	0	0	14	18	14
t_2	24	24	35	39	39
t_3	54	54	74	64	74
t_4	86	86	102	107	102
$Avail[1]$	14	74	102	102	102
$Avail[2]$	39	39	39	39	39

TABLE 6.2 – Récapitulatif des résultats obtenus sans prévision des disponibilités des ressources

	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$	$n=7$	$n=8$	$n=9$
$Avail[2]$	39	45	55	30	28	32	41	52	38
$\hat{Avail}[2], \alpha = 0.3$	40.25	39.87	41.41	45.48	40.84	36.98	35.49	37.14	41.60
$\hat{Avail}[2], \alpha = 0.5$	40.25	39.62	42.31	48.65	39.32	33.66	32.83	36.91	44.45

TABLE 6.3 – Récapitulatif des résultats obtenus avec prévision des disponibilités des ressources

exécutée par la ressource r_1. D'après l'exemple précédent, cette tâche peut être exécutée (commencée à être exécutée à l'instant $t = 14$). Or, la ressource r_1 si elle est en cours d'exécution d'autres tâches (i.e. il existe des tâches dans sa file d'attente) la date de début au plus tôt de la prochaine tâche à exécuter par cette ressource est déterminée en utilisant l'heuristique proposée précédemment pour la prévision de la disponibilité des ressources utilisées. Á titre d'exemple, soit 8 observations des disponibilités de la ressource r_1 pour exécuter la tâche t_1 données par le Tableau 6.3 avec la série lissées pour deux valeurs du paramètre α. Ainsi, la disponibilité de cette ressource pour exécuter t_1 est de 41.60 ou 44.45 pour respectivement α=0.3 et α=0.5. Pour des raison de simplicité, nous supposons que la disponibilité de cette ressource pour exécuter la tâche t_3 est une constante.

Un exemple de l'application de cette heuristique pour un ensemble d'observations des disponibilités de la ressource r_2 est donné par le tableau 6.3 pour $\alpha = 0.3$ et $\alpha = 0.5$. Ainsi, la date de début au plus tôt de la tâche t_1 par r_1 est de 41.60 (α=0.3). Après affectation de la tâche t_1 à la ressource r_1, on aura $avail[1] = 41.60 + 14 = 55.6$ et $AFT(t_1) = 55.6$.

Par conséquent, les nouvelles valeurs des quatre paramètres introduits précédemment, avec la prise en compte des prévisions des disponibilités des ressources, obtenues pour

	$EST(t_j, r_1)$	$EST(t_j, r_2)$	$EFT(t_j, r_1)$	$EFT(t_j, r_2)$	$AFT(t_j)$
t_1	41.6	0	55.6	18	55.6
t_2	65.6	65.5	76.6	85.6	85.6
t_3	95.6	95.6	115.6	105.6	105.6
t_4	116	117.6	132	138.6	132
$Avail[1]$	55.6	55.6	105.6	132	132
$Avail[2]$	0	85.6	85.6	85.6	85.6

TABLE 6.4 – Récapitulatif des résultats obtenus avec prévision des disponibilités des ressources

l'ordonnancement de cet exemple est donné par le tableau 6.4.

6.4.2 La fonction objectif du coût global d'exécution

Nous décrivons dans ce qui suit la fonction objectif du coût global d'exécution. Rappelons que dans ce chapitre deux types de ressources sont distinguées (machines virtuelles et ressources humaines). Sans perte de généralité, nous supposons dans ce qui suit que le coût engendré par l'utilisation d'une machine virtuelle pour exécuter une tâche qui nécessite l'intervention d'une ressource humaine est négligeable par rapport au coût engendré par l'utilisation de cette ressource humaine. Rappelons que le coût global est composé de deux quantités, à savoir :

1. Le coût d'exécution des tâches constituant le modèle de processus à exécuter.
2. Le coût de transfert engendré par les échanges de données entre les différentes ressources utilisées.

La fonction du coût global d'exécution notée *cost* peut être définie comme étant la somme des ces deux dernières quantités en considérant toutes les tâches de l'instance à exécuter. Formellement, elle est donnée par l'équation suivante :

$$cost = \sum_{j=1}^{n} \left\{ EC(r(t_j)) + \sum_{p \in pred(t_j)} TC(r(t_j), r(t_p)) \right\} \tag{6.12}$$

6.4.3 Position du problème

Une fois les différents paramètres pris en compte dans notre modèle et le deux fonctions objectifs (le temps global d'exécution et le coût global engendré par l'utilisation d'un

ensemble de ressources) décrites, le problème auquel nous nous intéressons dans ce chapitre peut être formulé comme suit. Soit une instance d'un processus métier et un ensemble de ressources (machines virtuelles et ressource humaines), l'objectif est de trouver un ordonnancement de façon à minimiser le temps et le coût d'exécution de cette instance. Rappelons que ce problème peut être abordé de plusieurs façons mais l'approche la plus appropriée est celle de l'optimisation bi-objectif. Notre objectif est donc de trouver un ordonnancement des différentes tâches constituant une instance d'un modèle de processus sur les ressources utilisées de façon à minimiser le temps global d'exécution et le coût engendré par l'utilisation de ces ressources. Formellement, ce problème peut être défini comme suit :

$\min\{makespan\}$ *et* $\min\{cost\}$, sous les contraintes :

$$\begin{cases} \text{contraintes de précédence du graphe de tâches (DAG)} & (1) \\ \text{prise en compte des paramètres définis précédemment} & (2) \\ \text{distinction de deux types de ressources (machines virtuelle et ressources humaines)} & (3) \\ \text{des ressources humaines peuvent être amenées à exécuter d'autres tâches n'appartenant pas à l'instance qui nous intéresse} & (4) \end{cases}$$

6.5 Les approches proposées

L'idée principale des approches décrites ci-après consiste dans un premier temps à relâcher le problème initial et dans un deuxième temps à recalculer les deux fonctions objectifs prises en compte en utilisant notamment l'heuristique décrite précédemment. Plus précisément, les approches consistent dans une première phase à trouver une affectation des tâches sans prise en compte de toutes les contraintes de précédence en utilisant trois stratégies complémentaires pour parcourir le graphe de tâches en question. Les solutions ainsi obtenues sont des bornes inférieures des valeurs exactes (temps global et coût global d'exécution). Le calcul par la suite de ces fonctions objectifs en tenant en compte des contraintes de précédence permet d'obtenir des bornes supérieures.

Les approches que l'on propose pour l'allocation de ressources et l'ordonnancement de tâches d'une instance d'un processus métier sont basées sur l'observation clé suivante.

Etant donnée un graphe de tâches et un ensemble de ressources hétérogènes. Utiliser une stratégie de parcours de haut en bas du graphe de tâches à exécuter (ce qui est souvent utilisé dans les algorithmes d'ordonnancement de graphe de tâches) ne donne pas souvent de meilleurs résultats par rapport à la stratégie qui consiste, par exemple, à parcourir le graphe en question de bas en haut.

Les deux premières approches que l'on propose (approche basée sur le coût global d'exécution et l'approche basée sur le temps global d'exécution) ont trois phases en commun, à savoir :

1. Phase de définition du graphe de tâches à exécuter : cette phase a pour objectif de définir un graphe de tâches sans cycle à exécuter représentant l'instance du modèle de processus en question. Les paramètres pris en compte dans cette phase, dans la définition du graphe de tâches à exécuter, sont ceux introduits dans la Section 6.2.
2. Phase de regroupement des tâches : le but de cette phase est de regrouper les tâches, constituant l'instance du modèle de processus en question, dans des ensembles distinct. Autrement dit, l'objectif de cette phase est de définir les niveaux du graphe de tâches en question.
3. Phase de sélection des solutions de Pareto (ou solutions non-dominées) : cette phase a pour objectif de sélectionner les solutions de Pareto parmi toutes les solutions obtenues.

La différence entre les trois approches proposées est au niveau de la phase d'allocation des ressources détaillée ci-après pour chacune de ces approches. C'est au niveau de cette phase que l'on fait appel à l'heuristique d'estimation de la disponibilité des ressources décrite précédemment. A l'issue des deux première phases (définition du graphe de tâches et regroupement des tâches dans des ensembles distincts) trois stratégies de parcours du graphe en question peuvent être utilisées, à savoir :

— (i) stratégie de parcours de haut en bas,
— (ii) stratégie de parcours de bas en haut
— et (iii) stratégie de parcours mixte.

6.5.1 Approche dirigée par la fonction du coût global d'exécution (*global execution cost driven-approach*)

Cette approche est guidée par la fonction objectif du coût global d'exécution. Nous décrivons dans ce qui suit les trois stratégies de parcours de l'instance du modèle de

processus en question. Il est important de noter que pour chacune des solutions (affectation des différentes tâches) obtenues les valeurs des deux fonctions objectifs prises en compte sont calculées afin de ne retenir que les solutions non-dominées.

(a) **Stratégie de parcours de haut en bas**

Cette stratégie consiste à commencer par l'affectation de la tâche initiale (t_{input}) à la ressource humaine ou machine virtuelle suivant sa nature qui minimise le coût d'exécution. Par conséquent, la ressource qui exécute la tâche t_{input} est déterminée par l'équation suivante :

$r(t_{input}) = r_s$ tel que :

$$EC(t_{input}, r_s) = \min_{r_j} EC(t_{input}, r_j) \tag{6.13}$$

Après cette affectation, le graphe est parcouru de haut en bas (du niveau 2 au niveau L). Une tâche qui appartient à un niveau $k \in \{2, ..., L\}$ est affectée à la ressource qui minimise le coût d'exécution en prennent en compte les coûts de transfert. Pour une tâche t_i la ressource qui l'exécute est donnée par l'équation suivante :

$\forall t_i \in l_k, r(t_i) = r_s$ tel que :
$EC(t_i, r_s) + \sum_{t_h \in pred(t_i)} TC(r(t_h), r(t_i)) =$

$$\min_{r_j} \left\{ EC(t_i, r_j) + \sum_{t_h \in pred(t_i)} TC(r(t_h), r(t_i)) \right\} \tag{6.14}$$

où $pred(t_i)$ est l'ensemble des prédécesseurs immédiats de la tâche t_i.

Les lignes 3 (avec $k = 1$) à 10 de l'Algorithme 6 correspondent à la stratégie de parcours de haut en bas.

(a) **Stratégie de parcours de bas en haut**

Pour cette stratégie, le graphe est parcouru de bas en haut. Par conséquent, nous commençons par le choix de la ressource qui minimise le coût d'exécution de la tâche t_{output}. La ressource $r(t_{output})$ est donnée par l'équation suivante :

$r(t_{output}) = r_s$ tel que :

$$EC(t_{output}, r_s) = \min_{r_j} EC(t_{output}, r_j) \tag{6.15}$$

Après cette affectation le graphe est parcouru de bas en haut en affectant les tâches appartenant au niveau $L - 1$ jusqu'au niveau 1. Pour un niveau k, ses tâches sont

Algorithm 6 Cost-based approach

read the DAG, the RG and associated attributes values ;
sort tasks at each level by traversing the DAG in a top-down fashion ; // let L be the set of levels and l_k the tasks // belonging to the level k
$k \leftarrow 1$; // first level
while $(k \leq L)$ **do**
for all tasks $t_i \in l_k$, compute $r(t_i)$ using equation 6.17
// assign task t_i to the resource $r(t_i)$
// that minimizes the execution cost
$mincost[k, t_i] \leftarrow r(t_i)$; // $mincost$ is a $L \times m$ matrix
// where line k corresponds to the assignment of all tasks
// obtained starting by the tasks assignement belonging to
// this level
$h \leftarrow k + 1$; // compute $r(t_i)$ for all tasks that belong to levels $h > k$
while $(h \leq L)$ **do**
for all tasks $t_i \in l_h$, compute $r(t_i)$ using equation 6.14
$mincost[k, t_i] \leftarrow r(t_i)$;
$h \leftarrow h + 1$
end while
$h \leftarrow k - 1$; // compute $r(t_i)$ for all tasks that belong // to levels $h < k$
while $(h \geq 1)$ **do**
for all tasks $t_i \in l_h$, compute $r(t_i)$ using equation 6.16
$mincost[k, t_i] \leftarrow r(t_i)$;
$h \leftarrow h - 1$
end while
$k \leftarrow k + 1$
endwhile
for each assignment, compute $AFT(t_{exit})$ using equations 6.8, 6.9 and 6.10 ;
select the Pareto solutions among L solutions ;

affectées aux ressources minimisant les coûts d'exécution et de communication. La ressource qui exécute une tâche t_i d'un niveau k est donnée par l'équation suivante :

$\forall t_i \in l_k, r(t_i) = r_s$ tel que :

$EC(t_i, r_s) + \sum_{t_h \in succ(t_i)} TC(r(t_i), r(t_h)) =$

$$\min_{r_j} \left\{ EC(t_i, r_j) + \sum_{t_h \in succ(t_i)} TC\left(r(t_i), r(t_h)\right) \right\} \tag{6.16}$$

où $succ(t_i)$ est l'ensemble des successeurs immédiats de la tâche t_i.

Les lignes 11 (avec $k = L$) à 15 de l'Algorithme 6 correspondent à la stratégie de parcours de bas en haut.

(a) **Stratégie de parcours mixte** La stratégie mixte, comme son nom l'indique utilise les deux stratégies précédentes simultanément, en commençant par l'affectation des tâches appartenant au niveaux intermédiaires $k \in \{2, ..., L-1\}$. Plus précisément, soit l_k l'ensemble des tâches par lesquelles on commence le processus d'affectation des ressources. Cette affectation est basée uniquement sur le coût d'exécution. Pour une tâche $t_i \in l_k$, son affectation est donnée par l'équation suivante :

$\forall t_i \in l_k, r(t_i) = r_k$ tel que :

$$EC(t_i, r_k) = \min_{r_j} EC(t_i, r_j) \tag{6.17}$$

L'affectation des tâches appartenant aux niveaux $k' < k$ et $k' > k$ est décidée en fonction des affectations précédentes. Ceci dit, les coûts de communication sont aussi pris en compte en plus de ceux d'exécution. Les affectations des tâches appartenant aux niveaux $k_1 < k$ et $k_2 > k$ sont déterminées en utilisant respectivement les stratégies de parcours de haut en bas et de bas en haut. Les Equations 6.14 et 6.16 sont ainsi appliquées de manière récurrente jusqu'à ce que toutes les tâches du processus en question soient affectées.

Les lignes 4 à 17 ($k \in \{2, ..., L-1\}$) de l' Algorithme 7 correspondent à la stratégie de parcours mixte.

6.5.2 Approche dirigée par la fonction du temps global d'exécution (*global execution time driven-approach*)

Tandis que l'approche précédente est dirigée par la fonction objectif du coût global d'exécution, cette approche est dirigée par la fonction objectif du temps global d'exécution. Étant donné que les ressources humaines peuvent être amenées à exécuter des tâches n'appartenant pas à l'instance du modèle de processus qui nous intéresse, l'appel à l'heuristique permettant de prédire leurs disponibilités est effectuée avant toute affectation.

De même que pour la précédente approche pour chacune des solutions obtenue les valeurs des deux fonctions objectif prises en compte sont calculées. Les trois approches de la phase d'allocation de ressources de cette approches sont décrites ci-après. Les trois stratégies de parcours du graphe de tâches en question suit le même schéma que la précédente approche. L'Algorithme 7 donne un aperçu de l'approche dirigée par la fonction du temps global d'exécution.

Les lignes 3 (avec $k = 1$) à 10 de l'Algorithme 7 correspondent à la stratégie de parcours de bas en haut. Les lignes 11 (avec $k = L$) à 15 correspondent à la stratégie de parcours de bas en haut. Les lignes 4-17 ($k \in \{2, ..., L-1\}$) correspondent à la stratégie de parcours mixte.

Par conséquent, si on commence l'affectation des ressources en exécutant en premier les tâches appartenant au niveau l_k, alors cette affectation est basée uniquement sur le temps d'exécution. Elle est donnée par l'équation suivante :

$\forall t_i \in l, r(t_i) = r_k$ tel que :

$$ET(t_i, r_k) = \min_{r_j} ET(t_i, r_j) \tag{6.18}$$

Pour les tâches appartenant aux niveaux l_{k+1} et l_{k-1}, les tâches sont affectées aux ressources disponibles en appliquant récursivement les deux équations suivantes :

$\forall t_i \in l_{k+1}, r(t_i) = r_k$ tel que :

$ET(t_i, r_k) + \sum_{t_h \in pred(t_i)} TT(r(t_h), r(t_i)) =$

$$\min_{r_j} \left\{ ET(t_i, r_j) + \sum_{t_h \in pred(t_i)} TT(r(t_h), r(t_i)) \right\} \tag{6.19}$$

$\forall t_i \in l_{k-1}, r(t_i) = r_k$ tel que :

$ET(t_i, r_k) + \sum_{t_h \in succ(t_i)} TT(r(t_i), r(t_h)) =$

$$\min_{r_j} \left\{ ET(t_i, r_j) + \sum_{t_h \in succ(t_i)} TT\left(r(t_i), r(t_h)\right) \right\} \tag{6.20}$$

Ces deux dernières équations sont appliquées jusqu'à ce que toutes les tâches soient exécutées.

Algorithm 7 Time-based approach

read the DAG, the RG and associated attributes values ;
sort tasks at each level by traversing the DAG in a top-down fashion ; // let L be the set of levels and l_k the tasks // belonging to the level k
$k \leftarrow 1$; // first level
while $(k \leq L)$ **do**
for all tasks $t_i \in l_k$, compute $r(t_i)$ using equation 6.18
// assign task t_k to the virtual machine $r(t_i)$
//that minimizes the execution time
$mintime[k, t_i] \leftarrow r(t_i)$; // *mincost* is a $L \times m$ matrix
// where line k corresponds to the assignment of all tasks
// obtained starting by the tasks assignement belonging to
// this level
$h \leftarrow k + 1$; // compute $r(t_i)$ for all tasks that belong // to levels $h > k$
while $(h \leq L)$ **do**
for all tasks $t_i \in l_h$, compute $r(t_i)$ using equation 6.19
$mintime[k, t_i] \leftarrow r(t_i)$;
$h \leftarrow h + 1$
end while
$h \leftarrow k - 1$; // compute $r(t_i)$ for all tasks that belong // to levels $h < k$
while $(h \geq 1)$ **do**
for all tasks $t_i \in l_h$, compute $r(t_i)$ using equation 6.20
$mintime[k, t_i] \leftarrow r(t_i)$;
$h \leftarrow h - 1$
end while
$k \leftarrow k + 1$
endwhile
for each assignment, compute *cost* using equation 6.12 ;
select the Pareto solutions among L solutions ;

6.5.3 Approche dirigée par la fonction du coût global et du temps global d'exécution (*global execution cost and time driven-approach*)

Cette approche est basée sur les deux premières approches. Elle consiste à exécuter les deux précédentes approches simultanément et de ne retenir que les solutions de Pareto. Elle est composée de deux étapes, à savoir :

1. La première étape consiste à exécuter les deux algorithmes décrits précédemment (approche basée sur le coût global d'exécution et approche basée sur le temps global d'exécution).
2. Sélection uniquement des solutions de Pareto.

6.6 Résultats numériques et discussions

Après avoir introduit les approches proposées, nous donnons dans cette section un récapitulatif des résultats numériques obtenus en utilisant les approches proposées pour l'allocation et l'ordonnancement des tâches d'un modèle de processus métier.

Nous commençons dans ce qui suit par décrire les données utilisées pour simuler l'exécution d'un modèle de processus métier dans le cadre du *Cloud computing*.

6.6.1 Données utilisées pour la simulation

Afin d'évaluer la qualité des solutions obtenues par les approches proposées dans ce chapitre, nous avons effectué une série de simulations. Le schéma de simulation utilisé consiste à générer de manière aléatoire des graphes sans cycle de tâches représentant l'instance à exécuter. Pour cela, nous considérons cinq familles de modèles de processus déterminées par le nombre de tâches qui les composent, à savoir $n \in \{50, 100, 300, 600, 1000\}$. Pour chaque instance (le nombre de tâches est fixé), nous définissons trois familles de modèles de processus appelées *Small*, *Medium* et *Large* que l'on note respectivement par S, M et L. Ces dernières sont définies en associant une probabilité pour l'existence d'une contrainte de précédence entre chaque paire de tâches t_i et t_j. Ces probabilités sont $p_s = 0.2$, $p_m = 0.4$ et $p_l = 0.6$ pour respectivement les familles S, M et L. Le nombre de machines virtuelles est fixé à $m = n/2$. Les autres paramètres du modèle que l'on propose pour l'allocation de ressources et l'ordonnancement de tâches d'un modèle de processus sont générés de manière aléatoire.

En ce qui concerne les disponibilités des ressources sollicitées et sans perte de généralité, nous supposons qu'elles suivent des lois exponentielles de paramètre λ [26]. Soit x la réalisation de la variable aléatoire représentant la disponibilité d'une ressource donnée. Pour générer une série de nombres aléatoires suivant une loi exponentielle de paramètre λ à partir d'une série de variables aléatoires suivant une loi uniforme dans l'intervalle $[0, 1]$, nous utilisons la formule suivante :

$$x = \frac{-1}{\lambda \log(u)} \tag{6.21}$$

6.6.2 Récapitulatif des résultats numériques et discussions

Nous donnons dans ce qui suit un récapitulatif des résultats numériques obtenus.

Nous commençons par comparer la qualité des solutions obtenues par les approches proposées en distinguant deux cas de figure : (i) le paramètre de lissage β, utilisé dans l'heuristique proposée pour l'estimation de la disponibilité des ressources, est calculé en utilisant la méthode des moindre carrées [27] (*estimated value*) et (ii) le paramètre de lissage est généré de manière aléatoire (*fixed value*). Le critère de comparaison utilisé est celui de l'erreur relative du temps global d'exécution de l'instance en question. Un récapitulatif des résultats numériques obtenues est donné par les figures 6.3, 6.4 et 6.5 en considérant les trois familles de modèles de processus décrites précédemment. Ces résultats montrent que l'on obtient de meilleurs résultats (erreur relative petite) lorsque l'on utilise un paramètre de lissage β estimé que lorsque l'on génère de manière aléatoire sa valeur et ce quelque soit la famille de modèle de processus exécutée. De plus, ils montrent que l'erreur relative ne dépasse pas 50% lorsque le paramètre de lissage est estimé en utilisant la méthode des moindres carrées.

Afin de montrer l'importance d'utiliser une heuristique pour prédire les valeurs futures des disponibilités des ressources utilisées, nous introduisons des fluctuations dans la disponibilité des ressources (i.e. les variables aléatoires modélisant les disponibilités des ressources ont un écart très important). L'objectif est de comparer l'erreur relative sur le temps global d'exécution lorsque l'on utilise l'heuristique proposée précédemment pour l'estimation de la disponibilité des ressources et l'utilisation de la moyenne des observations passées pour estimer la disponibilité d'une ressource donné. Un récapitulatif

26. La densité de probabilité d'une variable aléatoire de loi exponentielle est : $f(x) = \lambda e^{-\lambda x}$

27. Cette méthode consiste à déterminer le paramètre de lissage β tout en minimisant la somme des erreurs entre les valeurs exactes et les valeurs obtenues par lissage

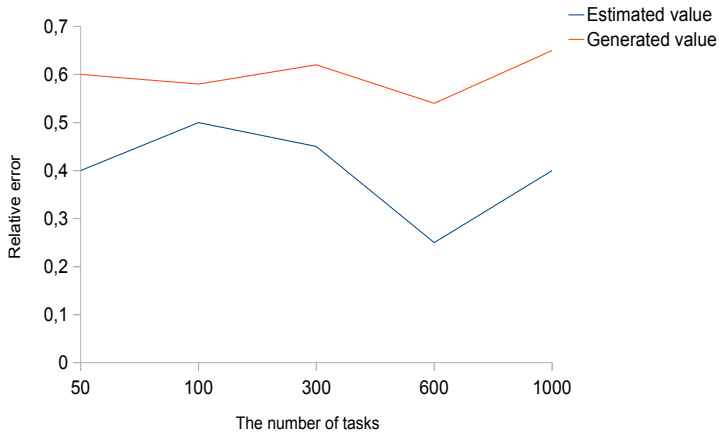

FIGURE 6.3 – Erreur relative sur le temps global d'exécution pour une famille de modèle de processus S (*Small*)

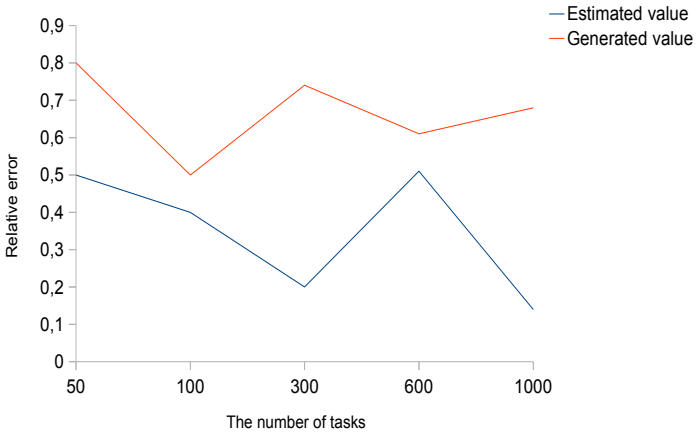

FIGURE 6.4 – Erreur relative sur le temps global d'exécution pour une famille de modèle de processus M (*Medium*)

des résultats numériques obtenus est donné par la figure 6.6. Ces résultats montrent deux choses :

1. Lorsque l'écart type (*Standard deviation*) est très petit (i.e. les valeurs des disponibilités sont très proches de la moyenne), l'utilisation de l'heuristique proposée ou le calcul de la moyenne des observations passées pour prédire les valeurs futures des disponibilités des ressources donnent presque les mêmes résultats. Dans ce cas, il est donc difficile d'affirmer que l'utilisation de l'heuristique proposée donne de meilleurs résultats que l'utilisation de la moyenne des observations passées.

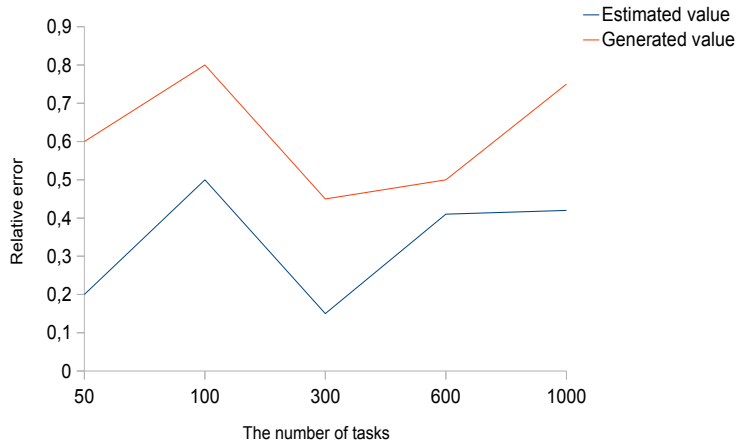

FIGURE 6.5 – Erreur relative sur le temps global d'exécution pour une famille de modèle de processus *L* (*Large*)

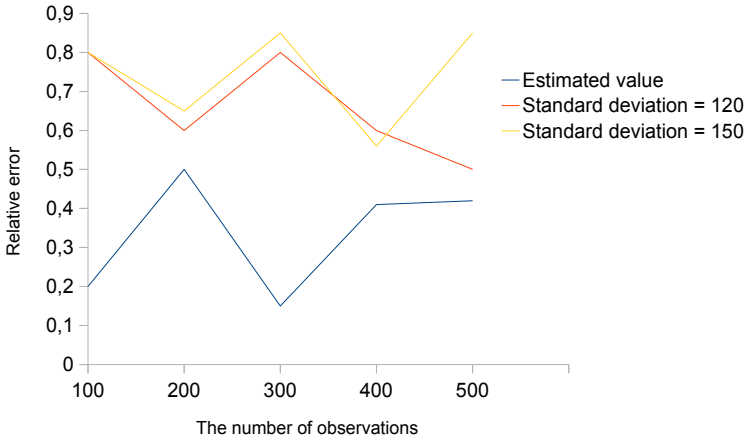

FIGURE 6.6 – Erreur relative sur le temps global d'exécution en fonction du nombre d'observations de la disponibilité des ressources

2. Lorsque l'écart type est très grand (i.e. des fluctuations importantes sont observées), alors l'utilisation de l'heuristique proposée pour l'estimation des disponibilités des ressources utilisées donne nettement de meilleurs résultats que lorsque la moyenne des observations est utilisée (dans ce cas toutes les observations ont un même poids).

6.7 Conclusion

Nous nous sommes intéressés dans ce chapitre au problème d'allocation de ressources et d'ordonnancement de tâches d'un processus métier. Contrairement au chapitre précédent où toutes les tâches sont supposées automatisées, nous avons distingué dans celui-ci deux types de ressources, à savoir machines virtuelles et ressources humaines. Ceci est justifié par le fait qu'il est difficile, voir impossible, d'automatiser toutes les tâches d'un processus métier donné. Le fait que le nombre de ressources humaines soit limité nécessite une gestion optimale de leurs files d'attente. De plus, ces ressources peuvent être amenées à exécuter d'autre tâches (qui n'appartiennent pas forcément au processus qui nous intéresse). Afin de prendre en compte ces deux derniers points, nous avons proposé respectivement une fonction objectif du temps global d'exécution d'une instance d'un processus (en tenant en compte le fait que le nombre de ressources humaines est limité) et l'utilisation des modèles de prévision afin d'estimer les disponibilités des ressources utilisées. De plus, une heuristique pour l'estimation de la disponibilité des ressources utilisées est proposée. Ainsi, nous avons montré l'importance d'utiliser une heuristique pour prédire la disponibilité des ressources notamment humaines.

Nous étudions dans le chapitre suivant le cas où plusieurs instances d'un même processus sont exécutés de manière concurrente. L'objectif visé, en plus des deux critères de qualité de service, à savoir le temps global d'exécution et le coût global d'exécution, est d'assurer le partage équitable des ressources utilisées entre les différentes instances prises en compte.

Chapitre 7

Optimisation bi-objectifs d'exécution de multiples instances d'un modèle de processus métier

Sommaire

7.1 Introduction

Nous avons proposé dans les deux chapitres précédents un ensemble de stratégies d'allocation de ressources et d'ordonnancement de graphe de tâches en distinguant deux cas de figure, à savoir : (1) le cas où toutes les tâches constituant le processus en question sont automatisées et (2) le cas où l'intervention de ressources humaines est nécessaire pour accomplir un sous ensemble de tâches du processus à exécuter (toutes les tâches ne sont donc pas automatisées). Dans les deux cas, nous nous sommes concentrés sur l'exécution d'une seule instance. Ces premières approches ne sont pas adaptées au cas où plusieurs instances sont exécutées simultanément, ce qui n'est pas rare lorsque l'on exécute un processus métier. En effet, généralement plusieurs cas du processus à exécuter sont instanciés et exécutés en parallèle. D'autre part, il est difficile, voire impossible, d'automatiser toutes les tâches en question. Il est par conséquent important de distinguer deux types de ressources lorsque plusieurs instances sont lancées de manière concurrente :

1. Ressources humaines supposées en nombre fini pour l'exécution des tâches non automatisées.
2. Machines virtuelles, comme dans les chapitres précédents, supposées en nombre infini modélisant ainsi la caractéristique d'« illusion de ressources infinies » du *Cloud*, pour l'exécution des tâches automatisées.

Les ressources disponibles sont donc intrinsèquement partagées par de nombreuses instances d'un processus métier. Par ailleurs, sur une plate-forme de ressources partagées les ordonnancements, proposés dans les deux chapitres précédents, peuvent avoir un impact négatif sur le temps de complétion des instances car l'exécution d'une des instances en question qui nécessite de nombreuses ressources peut être retardée et/ou elle-même retarder les autres instances qui accèdent de manière concurrente aux ressources disponibles. Il est par conséquent intéressant de proposer des approches d'allocation des ressources disponibles et d'ordonnancement des tâches d'un ensemble d'instances en prenant en compte un critère de partage équitable des ressources mises à la disposition des utilisateurs.

Un ordonnancement de plusieurs instances (DAGs) est équitable si une dégradation du temps global d'exécution (*makespan*) de chacune des instances exécutées en concurrence, par rapport à son temps global d'exécution si chaque instance disposait à elle seule de toutes les ressources disponibles, est la même pour toutes les instances.

Si plusieurs approches d'allocation de ressource et d'ordonnancement de tâches d'un seul graphe ont été proposées très peu d'études étudient le cas où plusieurs graphes s'exé-

cutent en parallèle [77][78]. De plus, ces études présentent plusieurs insuffisances les rendant inexploitables dans le cas que nous étudions dans ce chapitre. Ces insuffisances sont résumées par les points suivants :

1. Tous les graphes de tâches exécutés en parallèle sont identiques. Autrement dit, une seule instance du processus en question est instanciée en plusieurs exemplaires.
2. Un seule critère d'optimisation est pris en compte (*makespan*).
3. Toutes les instances exécutées arrivent en même temps dans le système.

Quand un ensemble d'instances (DAGs) d'un processus métier donné sont exécutées de manière concurrente quatre cas peuvent être distingués en fonction de la nature des tâches (automatisées ou pas) et du nombre d'instances à exécuter simultanément :

1. Toutes les tâches sont automatisées et une seule instance est exécutée.
2. Toutes les tâches sont automatisées et plusieurs instances sont à exécuter : si toutes les tâches des instances à exécuter sont automatisées et étant donné l'hypothèse sur le nombre de machines virtuelles mises à la disposition des utilisateurs, alors les instances sont exécutées de manière indépendante dans le sens où elles n'accèdent pas de manières concurrentes aux ressources. Autrement dit, dès qu'une tâche est affectée à une ressource une nouvelle machine virtuelle est instanciée. Par conséquent, les approches proposées pour l'allocation de ressources et l'ordonnancement des tâches d'un processus scientifique peuvent être utilisées.
3. Toutes les tâches ne sont pas automatisées (il existe un sous ensemble de tâches qui nécessitent l'intervention de ressources humaines pour les accomplir) et une seule instance est exécutée : dans ce cas l'approche décrite dans le chapitre précédent peut être appliquée.
4. Il existe un sous ensemble de tâches qui ne sont pas automatisées et plusieurs instances d'un processus métier donnée accèdent de manière concurrente aux ressources disponibles. Les approches proposées dans les deux chapitres précédent ne peuvent pas être appliquées dans ce cas.

Afin de mieux situer les contributions de ce chapitre par rapport à celles des deux précédents, un récapitulatif des quatre cas décrits précédemment est donné par le tableau 7.1.

Nous nous intéressons dans ce chapitre au dernier cas où toutes les tâches ne sont pas automatisées et plusieurs instances sont à exécuter.

Nombre d'instances / Nature des tâches	Une seule instance	plusieurs instances
Toutes les tâches automatisées	Chapitre 5	Chapitre 5
Un sous ensemble de tâches non automatisées	Chapitre 6	L'objet de ce chapitre

TABLE 7.1 – Approches proposées en fonction de la nature des tâches à exécuter et du nombre d'instances

En résumé, étant donné un ensemble d'instances d'un processus métier, la présence de plusieurs ressources pouvant exécuter une tâches donnée pose plusieurs challenges dont les plus significatifs sont résumés par les points suivants :

1. Quelle est la meilleure approche pour traiter du problème d'allocation de ressources et d'ordonnancement de tâches d'un ensemble d'instances d'un processus métier lorsque elles y accèdent de manière concurrente ?
2. Que devient la notion d'équité lorsque plusieurs critères d'optimisation sont pris en compte ?
3. Peut-on élaborer des algorithmes efficaces pour une allocation de ressources et un ordonnancement optimisé d'un ensemble d'instances d'un processus métier quand deux critères de qualité de services sont pris en compte ?

Nous abordons ces questions, dans ce chapitre, comme suit :

1. Nous proposons une formulation du problème d'allocation de ressources et d'ordonnancement de plusieurs instances d'un processus métier s'exécutant de manière concurrente. Le cadre de l'analyse est celui de l'ordonnancement de plusieurs graphes de tâches et la technique utilisée est celle de l'optimisation bi-objectifs.
2. Nous étendons la définition de l'équité (*fairness*) au cas où deux critères d'optimisation conflictuels sont pris en compte en distinguant deux cas, à savoir :
   a. Toutes les instances arrivent en même temps dans le système.
   b. Les instances arrivent au fur et à mesure dans le système.
3. Nous proposons un ensemble de stratégies d'allocation de ressources et d'ordonnancement des instances en question. L'objectif de ces stratégies est l'optimisation des deux critères cités précédemment et l'assurance d'un partage équitable des ressources disponibles.

Le reste de ce chapitre est organisé comme suit. La section suivante est consacrée à la présentation du modèle que l'on propose pour l'allocation de ressources et l'ordonnan-

cement d'un ensemble d'instances d'un processus métier. La Section 7.3 est consacrée à la présentation des approches optimisées que l'on propose pour une gestion optimale des ressources mises à la disposition des utilisateurs pour l'exécution d'un ensemble d'instances qui y accèdent de manière concurrente. Un récapitulatif des résultats numériques est donné par la Section 7.4. La Section 7.5 conclut le chapitre.

7.2 Formulation du problème d'allocation et d'ordonnancement d'instances d'un processus métier

7.2.1 Formulation du problème

Rappelons que l'objectif de ce chapitre est la proposition d'un ensemble de stratégies optimisées d'exécution d'un ensemble d'instances d'un processus métier tout en tenant en compte deux critères de qualité de service conflictuels, à savoir le temps et le coût global d'exécution. De plus, un troisième critère est pris en compte afin d'assurer le partage équitable des ressources mises à la disposition des utilisateurs. Comme mentionné auparavant, nous distinguons deux types de ressources, à savoir : (1) les ressources humaines et (2) les machines virtuelles. Si la tâche est automatisée et est affectée (exécutée donc sur une machine virtuelle) une image de la machine virtuelle type choisie est immédiatement instanciée. Si la tâche est affectée à une ressource humaine, son exécution ne peut commencer que lorsque cette dernière est libre.

Les tâches parallèles peuvent être classées en trois catégories, à savoir :

1. Les tâches dites *rigides* où le nombre de ressources qui les exécutent est fixé a priori.
2. Les tâches *modelables* où le nombre de ressources qui les exécutent est fixé juste avant l'exécution. Une fois, ce nombre de ressources fixé il reste inchangé tout au long de l'exécution des tâches en question.
3. Les tâches *malléables*, où le nombre de ressources les exécutant peut être changé même en cours de l'exécution.

Les processus métiers auxquels nous nous intéressons, dans ce chapitre, sont composés de tâches *rigides* où le nombre de ressources requises pour l'exécution d'une de ces tâches est égal à un. Autrement dit, chacune des tâches constituant le processus en question est exécutée par une et une seule ressource (machine virtuelle ou ressource humaine suivant sa nature).

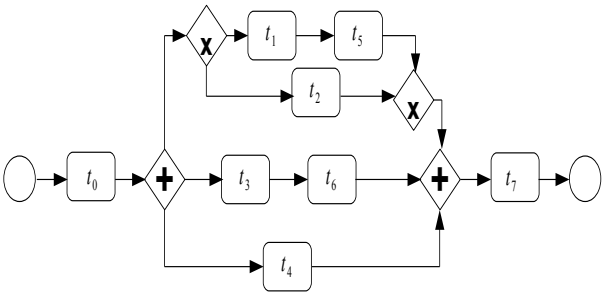

FIGURE 7.1 – Exemple d'un processus métier

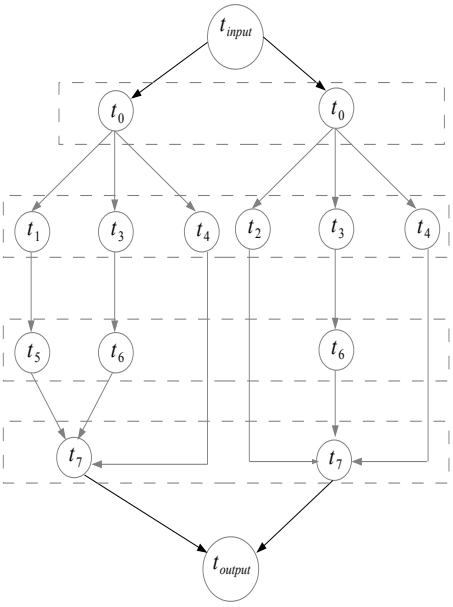

FIGURE 7.2 – Composition parallèle des instances du processus de la Figure 7.1

La figure 7.1 est une représentation d'un exemple d'un processus métier en utilisant les notations BPMN (Business Process Model Notation [93][101]). Il est composé de huit tâches notées $t_i, i = 0, ..., 7$ et de trois patrons de flux de contrôle (parallèle, séquence et choix exclusif). Deux instances sont donc possibles. La première instance consiste à exécuter toutes les tâches du processus sans t_2 et la deuxième instance consiste à exécuter toutes les tâches sans les tâches t_1 et t_5.

7.2.2 Description des fonctions objectifs prises en compte et du critère de l'équité

Nous décrivons dans ce qui suit les deux fonctions objectifs ainsi que le critère de l'équité pris en compte dans le modèle que l'on propose pour l'optimisation d'exécution d'un ensemble d'instances d'un modèle de processus métier. A partir d'une spécification d'un modèle de processus, un graphe de tâches sans cycle est construit et l'ensemble des instances possibles est déterminé en s'appuyant sur les différents patrons de contrôle et de données utilisés dans la définition du processus métier en question [86][87][88][89][90][91]. Nous donnons dans ce qui suit les définitions formelles des deux fonctions objectifs prises en compte et du critère de partage équitable des ressources. Pour le calcul de ces deux fonctions pour chacune des instances nous utilisons les mêmes paramètres que les deux chapitres précédents (le temps et le coût d'exécution, le temps et le coût de transfert). Les machines virtuelles mises à la disposition des utilisateurs sont en nombre infini et les ressources humaines disponibles sont en nombre fini.

Par ailleurs, il est important de noter que les données sont transmises uniquement via les machines virtuelles utilisées. Autrement dit, si une tâche donnée t_i est exécutée par une ressource humaine les données sont envoyées de la machine virtuelle $VM(t_i)$ à une (aux) autre(s) machine(s) virtuelle(s) suivant le nombre de successeurs de la tâche t_i. Dans ce cas, le temps d'exécution de la tâche t_i sur la machine $VM(t_i)$ est supposée négligeable par rapport à son temps d'exécution par la ressource humaine en question.

Fonction objectif du temps global d'exécution

Pour la fonction du temps global d'exécution, l'objectif consiste à minimiser le temps d'exécution de l'ensemble des instances exécutées de manière concurrente. Soit f cette fonction objectif définie par l'équation suivante :

$$f = \min_{1 \leq j \leq n} \Phi\{makespan(I_j)\} \tag{7.1}$$

Avec :

1. $makespan(I_j)$ est le temps global d'exécution de l'instance I_j.
2. Φ est une fonction d'agrégation des temps d'exécutions des instances prises en compte. Cette fonction est discutée ci-après.

Fonction objectif du coût global d'exécution

Pour la fonction du coût d'exécution, l'objectif est de minimiser le coût d'exécution de l'ensemble des instances prises en compte. Soit g cette fonction objectif définie par l'équation suivante :

$$g = \min_{1 \leq j \leq n} \Phi\{cost(I_j)\} \tag{7.2}$$

Avec :

1. $cost(I_j)$ est le coût global d'exécution de l'instance I_j.
2. Φ est une fonction d'agrégation des coûts d'exécutions des instances prises en compte. Cette fonction est discutée ci-après.

Critère de partage équitable des ressources disponibles (équité)

Dès lors que toutes les tâches ne sont pas automatisées et que l'intervention de ressources humaines est nécessaire en nombre fini, et que plusieurs instances accèdent de manière concurrente aux ressources disponibles il est plus que jamais indispensable de tenir compte du partage équitable de ces ressources. Nous présentons, dans ce qui suit, un critère pour assurer un partage équitable des ressources disponibles tout en tenant compte des deux fonctions objectifs définies précédemment, à savoir le temps global d'exécution et le coût global d'exécution.

La notion d'équité (*fairness*) a été utilisée de différentes façons [151][152][153][154]. Cependant, elle se réfère toujours au cas de l'optimisation mono-critère (minimisation du temps d'exécution). Nous l'utilisons dans ce chapitre dans le sens où si une instance est exécutée sans partage de ressources, son temps d'exécution en cas de partage avec d'autres instances sera forcément supérieur ou égal à son temps d'exécution lorsque les ressources ne sont pas partagées. Ainsi, nous définissons « un facteur de ralentissement » (*slowdown factor*) qui mesure l'impact qu'aurait le fait de partager un ensemble de ressources par un ensemble d'instances. Soient I_j une des instances de l'ensemble I, un ensemble de ressources et un ordonnancement (toutes les tâches sont affectées), alors la valeur du facteur de ralentissement de l'instance I_j en se référant au temps global d'exécution noté $slowdown_{time}(I_j)$ est donnée par l'équation suivante :

$$slowdown_{time}(I_j) = \frac{FT_s^{I_j}(t_{exit})}{FT_m^{I_j}(t_{exit})}$$

Où :

1. $FT_s^{I_j}$ est le temps global d'exécution de l'instance I_j quand toutes les ressources sont mises à disposition et à elle seule. Autrement dit, l'ensemble des instances à exécuter est réduit à l'instance I_j.

2. $FT_m^{I_j}(t_{exit})$ est le temps global d'exécution de l'instance I_j lorsque les ressources utilisées sont éventuellement partagées par d'autres instances de l'ensemble I.

En ce qui concerne le coût global engendré par l'utilisation d'un ensemble de ressources et afin de définir les priorités des différentes instances à exécuter, nous définissons le coût cumulé *ccost*. Notons que contrairement à la fonction objectif du temps global d'exécution, le coût global d'exécution d'une instance donnée n'est pas forcément plus élevé quand les ressources sont partagées par plusieurs instances. Ceci est une conséquence directe du fait que les approches que l'on propose sont des heuristiques et non pas des algorithmes exacts. La valeur du coût global cumulé (augmentation ou diminution étant donné que les approches proposées sont des heuristiques) pour une instance I_j et un ordonnancement donné est défini par l'équation suivante :

$$ccost(I_j) = \frac{cost_s^{I_j}}{cost_m^{I_j}} \tag{7.3}$$

Où :

1. $cost_s^{I_j}$ est le coût global d'exécution de l'instance I_j quand toutes les ressources sont mises à sa disposition et à elle seule. C'est-à-dire l'ensemble $I = \{I_j\}$.

2. $cost_m^{I_j}$ est le coût global d'exécution de l'instance I_j lorsque les ressources utilisées pour l'exécuter sont éventuellement utilisées par d'autres instances de l'ensemble I.

Le problème auquel nous nous sommes intéressés dans ce chapitre peut être formulé comme suit : étant donné un ensemble d'instances I et un ensemble de ressources. L'objectif est de déterminer une allocation des ressources disponibles pour l'exécution des instances en question tout en minimisant les temps et les coûts globaux des instances de l'ensemble I et en assurant un partage équitable de ces ressources. Soient *makespan(I_j)* et *cost(I_j)* le temps global et le coût global d'exécution de l'instance I_j. Formellement, le problème auquel nous nous sommes intéressés peut être défini comme suit.

$$\begin{cases} \min_{1 \leq j \leq n} \Phi\{makespan(I_j)\} \\ \min_{1 \leq j \leq n} \Phi\{cost(I_j)\} \\ \min fairness \end{cases} \tag{7.4}$$

Comme mentionné auparavant, ce problème peut être abordé de plusieurs façons. Nous avons opté pour une approche multi-objectifs et la notion d'optimalité utilisée est celle de Pareto introduite précédemment. Plus précisément, nous proposons dans ce qui suit trois stratégies d'allocation de ressources et d'ordonnancement des instances d'un processus métier.

7.3 Approches optimisées pour l'exécution concurrente d'instances d'un processus métier

Rappelons que l'objectif de ce chapitre est la proposition d'un ensemble de stratégies pour l'allocation de ressources et l'ordonnancement de tâches d'un ensemble d'instances d'un processus métier.

Comme mentionné auparavant, en plus des deux critères de qualité de service, à savoir le temps et le coût d'exécution, un troisième critère doit être pris en compte lorsque plusieurs instances d'un processus métier accèdent de manière concurrente aux ressources disponibles. Il concerne le partage équitable des ressources disponibles. Les approches que l'on propose ci-après sont composées de deux étapes :

1. La première étape consiste à appliquer l'approche présentée dans le chapitre précédent sur chacune des instances à exécuter et ne retenir que les solutions non dominées. Autrement dit, dans une première étape l'affectation des tâches se fait en supposant qu'il n'y pas de partage de ressources entre elles.
2. La deuxième étape consiste à raffiner chacune des solutions obtenues à l'issue de la première étape en tenant compte du fait que les ressources sont partagées par un ensemble d'instances tout en assurant un partage équitable des ressources utilisées.

Lorsque plusieurs instances accèdent de manière concurrente à un ensemble de ressources deux cas de figure peuvent être distingués :

1. Le cas où toutes les instances arrivent en même temps dans le système. Autrement

dit, avant de commencer l'exécution nous connaissons exactement le nombre et les types d'instances à exécuter.

2. Le cas où les instances arrivent au fur et à mesure dans le système. Dans ce cas, ne nous connaissons pas a priori le nombre d'instances à exécuter. Le processus des arrivées de ces instances peut donc être modélisé par une variable aléatoire.

Nous décrivons dans ce qui suit trois stratégies pour l'allocation de ressources et d'ordonnancement d'un ensemble d'instances d'un processus métier. Rappelons que notre objectif est de minimiser le temps global et le coût global engendrés par l'utilisation d'un ensemble de ressources disponibles tout en assurant un partage équitable de ces dernières. L'idée principale des stratégies proposées est de définir un niveau de priorité pour chacune des instances prises en compte. Il est important de noter que cette priorité évolue dans le temps dans le sens où l'exécution des tâches se fait niveau par niveau des graphes modélisant l'ensemble des instances à exécuter simultanément. Le schéma général de l'approche présentée est constitué de trois phases :

1. Phase de composition des instances à exécuter.
2. Phase d'allocation des ressources et d'ordonnancement des tâches.
3. Phase de sélection des solutions non-dominées (Pareto).

Un aperçu de l'approche proposée est donné par l'Algorithme 8.

7.3.1 Phase de composition des instances

Une solution triviale pour exécuter un ensemble d'instances d'un processus donné est de les ordonnancer de manière séquentielle. C'est-à-dire exécuter une instance après l'autre dans le sens où une instance ne peut être exécutée que lorsque celle qui la précède est accomplie. Le problème majeur de ce type de composition est la non utilisation efficace des ressources. En effet, si par exemple deux instances doivent être exécutées et que la ressource r n'exécute que la première tâche, cette dernière sera libre (elle n'exécute aucune tâche) jusqu'à ce que cette première instance soit terminée. De plus, les temps d'exécution des instances qui ne seront pas exécutées dès le début du processus d'allocation des ressources seront très élevés.

Pour éviter cet inconvénient, une autre alternative consiste à exécuter toutes les instances de manière parallèle (concurrente). Il existe plusieurs façons d'exécuter un ensemble d'instances de manière concurrente. Nous proposons dans ce qui suit une construction d'un

Algorithm 8 Approche basée sur le partage équitable des ressources

1: Read the DAGs, the RG and associated attributes values ;
2: Run each instance (DAG) using time-based approach and cost-based approach. Store the finish time and the cumulative cost of each task in descending order
3: Mark each instance as unexecuted let A to be the set on these instances, and mark every level in each instance as unexecuted. Set the slowdown and the cumulative values as 0, and sort instances in descending order of their overall execution time and overall execution cost.
4: **while** (there are unexecuted instances) **do**
5: $I \leftarrow$ first instances in unexecuted set A
6: $l \leftarrow$ the first ready level that has not been executed on I
7: Apply the strategy allocation resources of the time-based algorithm
8: **if** (l is the last level of instance I) **then**
remove I from A
9: **else**
10: **for** each allocation strategy **do**
compute the slowdown factor and the cumulative cost for each level
sort A in ascending order of the slowdown factor and the cumulative cost using Equation 7.11 and 7.12
11: **if** (two instances have the same rank) **then**
12: compare the remaining overall execution cost and the remaining incurred cost using the same order as in the previous step
13: **end if**
14: **end for**
15: **end if**
16: **end while**
17: Select the Pareto solutions among the obtained solutions ;

graphe de tâches en combinant toutes les instances en question. Autrement dit, cette composition permet de créer une seule instance (graphe) en ajoutant une tâche d'entrée qui sera un prédécesseur immédiat de toutes les tâches t_{input} des instances à exécuter. Une tâche de sortie est aussi ajoutée constituant le successeur immédiat de toutes les tâches t_{output}. A titre d'exemple, le graphe obtenu pour les deux instances de la Figure 7.1 est donné par la Figure 7.2. Notons que les temps (exécution et communication) et les coûts (exécution et communication) des deux tâches ajoutées sont nuls. Nous avons dans ce

chapitre opté pour ce deuxième type de composition.

Une fois que la composition des instances est construite, une façon naturelle de les exécuter est d'utiliser une des approches proposées dans les chapitres précédents selon que les tâches soient toutes automatisées ou pas, et en fonction du nombre d'instances à exécuter. Cependant, ceci ne garantit pas le partage équitable des ressources.

Pour assurer l'équité recherchée, nous raffinons les solutions générées par nos précédentes approches en utilisant les trois stratégies décrites ci-après.

7.3.2 Phase d'allocation de ressources et d'ordonnancement de tâches

Pour faciliter la compréhension des stratégies proposées et sans perte de généralité, soient deux instances I_i et I_j à exécuter (qui s'exécutent de manière concurrente). Pour décider des tâches appartenant à un niveau donné à exécuter dans la prochaine étape de notre approche, nous proposons trois stratégies pour calculer leur priorité :

1. Moyenne des temps de fin au plus tôt et du coût cumulé : nous appelons cette stratégie *average*. Elle est basée, comme son nom l'indique, sur le calcul de la moyenne des temps de fin au plus tôt des tâches appartenant au niveau en question et de la fonction du coût cumulé depuis le début de l'exécution de l'instance en question. Par conséquent le facteur de ralentissement (*slowdown factor*) de l'instance en question et du coût cumulé (*ccost*) sont donnés par les deux équations suivantes :

$$slowdown(I_j) = \frac{\sum_{t_i \in l_k} slowdown(t_i)}{|l_k|} \tag{7.5}$$

$$ccost(I_j) = \frac{\sum_{t_i \in l_k} ccost(t_i)}{|l_k|} \tag{7.6}$$

2. Meilleur temps de fin au plus tôt et coût cumulé : nous appelons cette stratégie *best*. Elle est basée sur la meilleure valeur du temps de fin au plus tôt des tâches appartenant au niveau du graphe en question et du coût cumulé depuis le début de l'exécution de l'instance en question. Par conséquent le facteur de ralentissement (*slowdown factor*) de l'instance en question et du coût cumulé *ccost* sont donnés par les deux équations suivantes :

$$slowdown(I_j) = \min_{t_i \in l_k} \{slowdown(t_i)\} \tag{7.7}$$

$$ccost(I_j) = \min_{t_i \in l_k} \{ccost(t_i)\} \tag{7.8}$$

3. Pire temps de fin au plus tôt et le coût cumulé : nous appelons cette stratégie *worst*. Elle est basée, comme son nom l'indique, sur la plus mauvaise valeur du temps de fin au plus tard des tâches du niveau concerné et du coût cumulé depuis le début de l'exécution de l'instance en question. Par conséquent le facteur de ralentissement *slowdown factor* de l'instance en question et du cout cumulé *ccost* sont donnés par les deux équations suivantes :

$$slowdown(I_j) = \max_{t_i \in l_k} \{slowdown(t_i)\} \tag{7.9}$$

$$ccost(I_j) = \max_{t_i \in l_k} \{ccost(t_i)\} \tag{7.10}$$

Ces attributs permettent de calculer, à chaque étape de l'approche que l'on propose, la priorité de chacune des instances exécutées de manière concurrente et de prendre ainsi la décision du prochain niveau à exécuter. Plus précisément, le prochain niveau à exécuter est basé sur deux métriques, à savoir le facteur de ralentissement (*slowdownfactor*) et la fonction du coût cumulé depuis le début de l'exécution. Une fois les tâches d'un niveau exécutées, l'approche proposée maintient une liste de tâches (de niveau) et actualise les valeurs des deux métriques *slowdownfactor* et *ccost*. Le prochain niveau d'une instance à exécuter est celle qui a la plus petite valeur du facteur de ralentissement et la plus grande valeur de la fonction du coût cumulé. Formellement, chaque instance I_j est caractérisée par deux valeurs ($slowdownfactor(I_j)$, $ccost\ (I_j)$) et cette instance (I_j) est plus prioritaire que l'instance I_i si et seulement si ces deux conditions sont vérifiées :

$$slowdown(I_j) < slowdown(I_i) \tag{7.11}$$

et,

$$cost(I_j) > cost(I_i) \tag{7.12}$$

Si l'équation 7.11 n'est pas vérifiée, alors on utilise le temps qui reste pour finir l'exécution de cette instance lorsque toutes les ressources sont à sa disposition. Plus précisément, nous calculons la différence entre le temps de fin de la tâche t_{exit} et le temps qui s'est écoulé depuis le début de l'exécution de cette instances. Formellement, le temps qui reste pour accomplir une instance I_j noté par $R(I_j)$ est donné par l'équation suivante :

$$R(I_j) = FT_s^{I_j}(t_{exit}) - FT_m^{I_j}(l_k) \tag{7.13}$$

Rappelons que $FT_s^{I_j}(t_{exit})$ est le temps global de l'exécution de l'instance I_j lorsque toutes les ressources disponibles sont à sa disposition (donc pas de partage de ressources). La quantité $FT_s^{I_j}(l_k)$ représente le temps qui s'est écoulé depuis le début de l'exécution de l'instance I_j sachant que l'on a exécuté les tâches appartenant au niveau k. Cette dernière est donnée par l'équation suivante :

$$FT_s^{I_j}(l_k) = \max_{t_i \in l_k}(FT_s(t_i)) \tag{7.14}$$

7.3.3 Phase de sélection des solutions de Pareto

L'objectif de cette phase est de sélectionner les solutions dites non dominées (Pareto). Plus précisément, étant donné qu'il y a trois stratégies de calcul des priorités, alors trois solutions différentes sont obtenues, et parmi elles seules celles qui ne sont pas dominées sont retenues.

Dans le cas où toutes les instances n'arrivent pas en même temps dans le système. C'es-à-dire, arrivent au fur et à mesure, les deux différences principales par rapport au premier cas sont données ci-après :

1. Le graphe de composition est actualisé à chaque arrivée d'une nouvelle instance dans le système.
2. En plus des deux contraintes données par les équations 7.11 et 7.12, une troisième contrainte doit être vérifiée. Elle concerne l'âge de l'instance en question. Formellement, soit $age(I_j)$ et $age(I_i)$ l'âge des instance I_j et I_i respectivement. Par conséquent, l'instance I_j est plus prioritaire que l'instance I_i si et seulement si les deux équations 7.11 et 7.12 sont vérifiées ainsi que la contrainte suivante :

$$age(I_j) > age(I_i) \tag{7.15}$$

7.4 Résultats numériques et discussions

Dans le but d'évaluer la qualité des solutions obtenues, en utilisant les stratégies l'allocation de ressources et d'ordonnancement d'instances de processus métiers exécutées de manière concurrente, nous présentons dans cette section un récapitulatif des résultats

numériques obtenus. Pour évaluer les trois stratégies décrites précédemment, une série de simulations ont été effectuées en utilisant le même schéma que dans les deux chapitres précédents. Plus précisément, nous discutons des valeurs de la non équité (*unfairness*) ainsi que des valeurs des deux fonctions objectifs prises en compte (le temps et le coût d'exécution).

Pour notre expérimentation, nous simulons un environnement de *Cloud Computing* en considérant cinq familles d'instances d'un processus scientifique. Chaque famille est associée au nombre de tâches qui la composent, c'est-à-dire $n \in \{50, 100, 300, 600, 1000\}$. Chaque famille d'instance est divisée en trois catégories en fonction de la densité de ses contraintes de précédence. Nous appelons ces familles $Small$, $Medium$ et $Large$, notées respectivement S, M et L. Ainsi, pour chaque paire de tâches (t_i, t_j) une contrainte de précédence les lie avec une certaine probabilité notée p tel que le graphe obtenu soit sans cycle (DAG). Les probabilités correspondantes pour les séries S, M et L sont respectivement égales à $p_S = 0.2$, $p_M = 0.4$ et $p_L = 0.6$. Le nombre de ressources noté m (machines virtuelles) est défini en fonction du nombre de tâches et sans perte de généralité fixé à $m = n/2$. Nous supposons qu'une tâche du processus en question peut être exécutée par toutes les ressources disponibles. Autrement dit, une tâche t_i du processus en question peut être affectée et exécutée sur un des types de machines virtuelles mis à la disposition de l'utilisateur si la tâche t_i est automatisée. Sinon, elle est affectée à l'une des ressources humaines disponible. Pour toutes les familles d'instances définies précédemment, le temps d'exécution est généré de manière aléatoire et uniformément distribué. Autrement dit, le temps d'exécution des tâches sur les ressources suit une loi de probabilité uniforme dont les valeurs varient entre un et dix. Le coût d'exécution des tâches sur les machines virtuelles suit aussi une loi uniforme définie sur l'intervalle $[0.1, 0.9]$. La bande passante entre les machine virtuelle suit une loi de probabilité uniforme définie sur l'intervalle $[1, 9]$ et la matrice des transferts de données est définie sur l'intervalle $[10, 90]$. Finalement, le coût d'entrée et de sortie des machines virtuelles utilisées suit une loi uniforme définie sur l'intervalle $[1, 9]$. Chaque série de simulation consiste à générer 1000 instances de chaque famille.

Comme dans la section consacrée à la présentation des approches proposées, nous distinguons dans ce qui suit deux cas de figure, à savoir : (1) toutes les instances arrivent en même temps et (2) les instances arrivent au fur et à mesure dans les système.

7.4.1 Les instances arrivent en même dans le système

Dans un premier temps, nous comparons les performances des trois stratégies présentées précédemment en utilisant trois métriques [148] :

1. Le pourcentage moyen de dégradation de la meilleure valeur (the Average Percentage Degradation from the best value (APD)) : elle représente le pourcentage moyen de la dégradation de la meilleure valeur obtenue par une stratégie donnée comparée à la meilleure solution obtenue par la même stratégie.
2. Le hombre de meilleures solutions *the best obtained value and (ii) the Number of Best Solutions (NBS)* : elle représente le nombre de fois où une stratégie donnée fournit la meilleure solution (temps ou coût minimum) en la comparant à toutes les solutions obtenues par les autres stratégies.

Notons que ces deux métriques sont appliquées aussi bien pour le critère temps d'exécution que pour le coût d'exécution.

Les pourcentages moyens de dégradation de la meilleure valeur (ADP) en ce qui concerne le temps et le coût d'exécution sont donnés par les Figure 7.3 et 7.4 pour respectivement la fonction objectif du temps d'exécution et du coût d'exécution. Les classements des trois stratégies proposées de la meilleure à la moins bonne de la Figure 7.4 sont donnés par :

— Pour $n = 50$, le classement est : meilleure (*best*), moyenne (*average*) et pire (*worst*).
— Pour $n = 100$, le classement est : pire (*worst*), moyenne (*average*) et meilleure (*best*).
— Pour $n = 600$, le classement est : pire (*worst*), moyenne (*average*) et meilleure (*best*).

De ce récapitulatif des résultats obtenus nous pouvons conclure qu'aucune des stratégie ne domine (donne de meilleure résultats) les deux autres. Notons que les mêmes résultats ont été obtenus lorsque l'on s'intéresse au nombre de solutions de Pareto générées par chacune des stratégies proposées. Cependant, il est important que lorsque l'écart entre les temps d'exécution (coût d'exécution) et les différentes ressources est petit, la stratégie qui consiste à utiliser la moyenne des temps de fin des tâches appartenant à un niveau donné donne de meilleurs résultas que les deux autres stratégies. En conclusion, étant donné qu'aucune des stratégies ne domine les deux autres, il est difficile d'en recommander une seule à un utilisateur. Autrement dit, nous recommandons aux utilisateurs d'utiliser les trois stratégies en ne choisissant les meilleures solutions parmi toutes les solutions

Stratégie d'allocation	Moyenne (Average)	Meilleure (Best)	Pire (Worst)
Valeurs NBS (temps, coût)	(14, 18)	(30, 13)	(21, 10)

TABLE 7.2 – Valeurs du critère NBS pour les trois stratégies proposées (Average, Best et Wors)

obtenues.

Le nombre de fois où chacune des stratégies proposées génère une meilleure solution que les deux autres stratégies pour les deux fonctions objectifs est donné par le tableau 7.2 suivant.

Nous avons remarqué que lorsque les deux critères sont pris simultanément en compte aucune des stratégies ne domine l'autre. Par conséquent, l'approche bi-objectif est l'approche la plus appropriée pour traiter le problème d'allocation de ressources et d'ordonnancement de tâches de plusieurs instances d'un processus métier exécutées de manière concurrente.

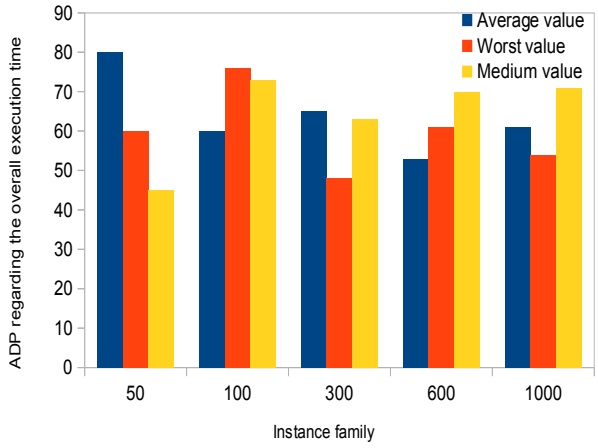

FIGURE 7.3 – Comparaison des trois stratégies sur le pourcentage moyen de dégradation de la meilleure valeur (critère temps)

En ce qui concerne le critère de la non équité de partage des ressources disponibles (*unfairness*), un récapitulatif des résultats numériques obtenus est donné par les figures 7.8, 7.9 et 7.10 comparant les trois stratégies proposées précédemment. Les figures 7.5, 7.6 et 7.7 représentent un récapitulatif des résultats obtenus en fonction du nombre de tâches d'une instance donnée et en prenant en compte les trois familles d'instances *small*, *medium* et *large*. D'après ce récapitulatif, nous pouvons conclure qu'aucune des stratégies

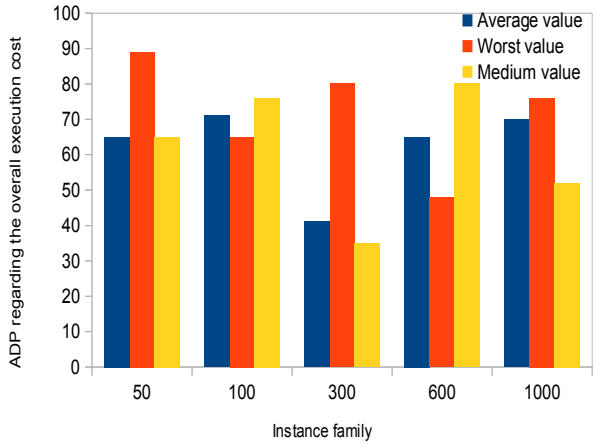

FIGURE 7.4 – Comparaison des trois stratégies sur le pourcentage moyen de dégradation de la meilleure valeur (critère coût)

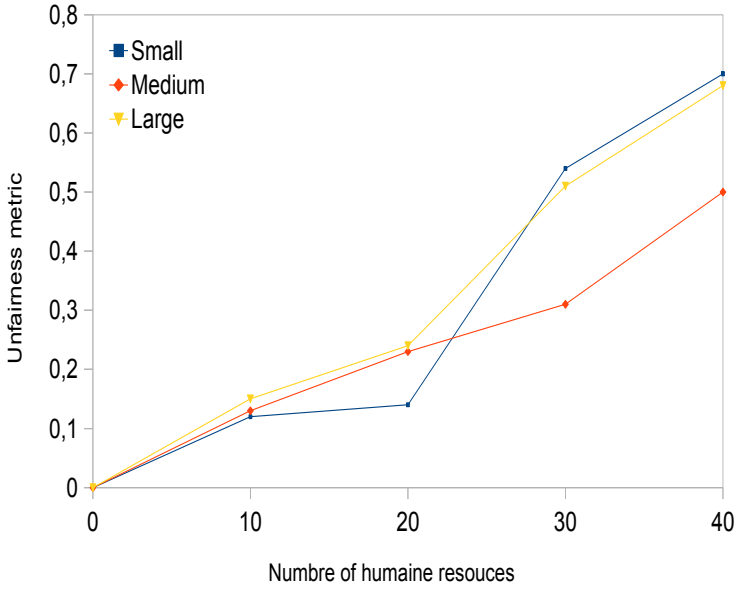

FIGURE 7.5 – Le critère de non équité (*unfairness*) pour chaque famille d'instances avec $n = 50$

ne donne de meilleurs résultats que les deux autres. Par conséquent, nous recommandons aux utilisateurs l'utilisation des trois stratégies simultanément et d'en retenir celles qui offrent les meilleurs résultats.

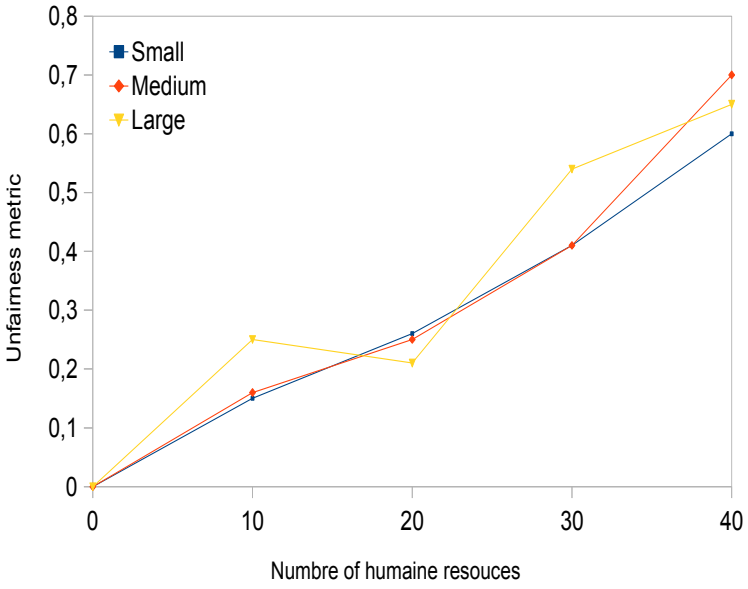

FIGURE 7.6 – Le critère de non équité (*unfairness*) pour chaque famille d'instances avec $n = 300$

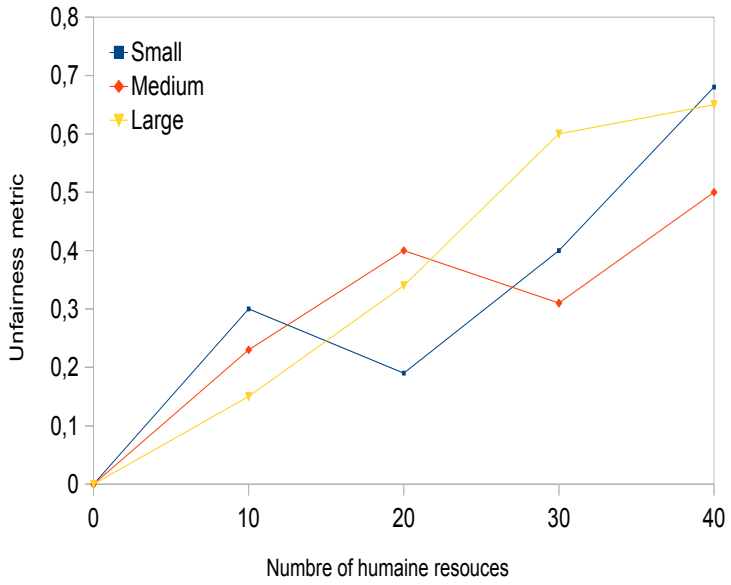

FIGURE 7.7 – Le critère de non équité (*unfairness*) pour chaque famille d'instances avec $n = 1000$

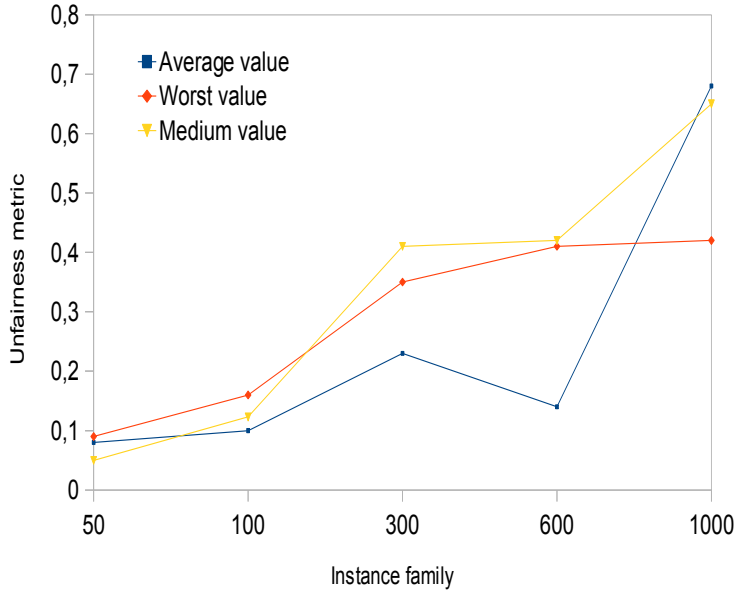

FIGURE 7.8 – Le critère de non équité en comparant les trois approches proposées dans la phrase d'allocation de ressources en fonction du nombre de tâches composant un instance

7.4.2 Les instances arrivent de manière aléatoire dans le système

Dans le cas où les instances d'un processus donné arrivent de manière aléatoire, nous utilisons le même schéma de simulation que dans le premier cas (c'est-à-dire toutes les instances arrivent en même temps dans le système). Le processus des arrivées des instances dans le système est supposé suivre une loi de Poisson. Un récapitulatif des résultats numériques obtenus est donné par la Figure 7.7. Cette dernière montre l'évolution du non partage équitable (*unfairness*) en fonction du nombre de ressources humaines mises à la disposition des utilisateurs. Ces résultats montrent deux choses :

1. Si le nombre de ressources humaines est nul (toutes les tâches sont automatisées), le critère *unfairness* égal à zéro. Autrement dit, aucune instance n'en retarde une autre à cause du fait que le nombre de machine virtuelle est supposé infini.
2. La valeur du critère *unfairness* augmente de manière logarithmique par rapport au nombre d'instances exécutées simultanément.

De cette dernière remarque, on en conclut qu'une idée intéressante afin de réduire la valeur du critère *unfairness* est de limiter le nombre de tâches qui peuvent être exécutées par une ressource humaine donnée.

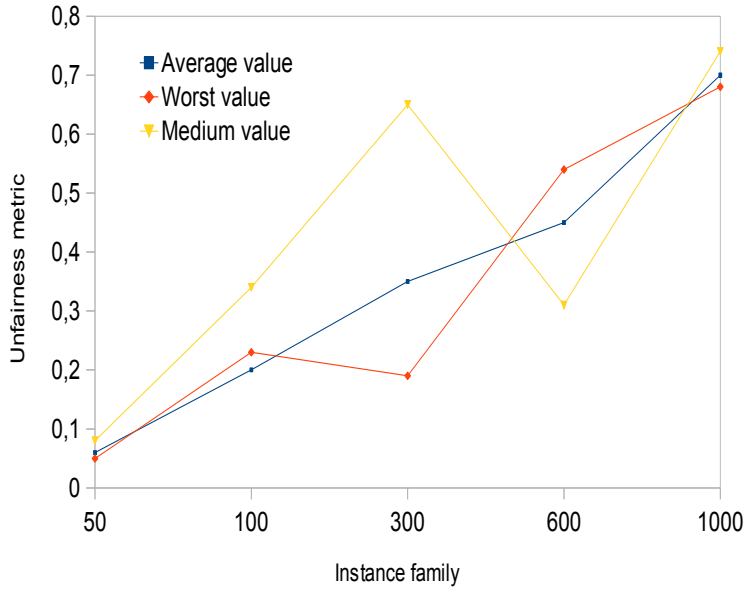

FIGURE 7.9 – Le critère de non équité en comparant les trois approches proposées dans la phrase d'allocation de rescousses en fonction du nombre de tâches composant un instance

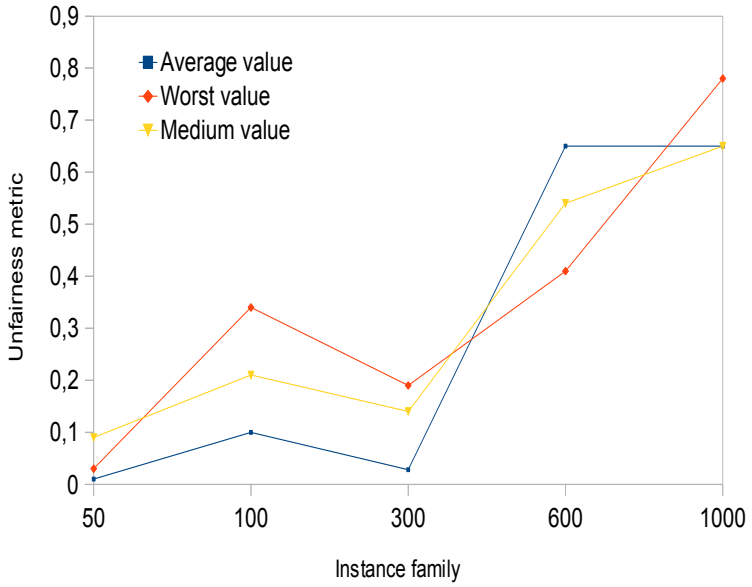

FIGURE 7.10 – Le critère de non équité en comparant les trois approches proposées dans la phrase d'allocation de rescousses en fonction du nombre de tâches composant un instance

7.5 Conclusion

Nous avons proposé dans ce chapitre des stratégies d'allocation de ressources et d'ordonnancement d'instances d'un processus métier. Plus précisément, un ensemble de stratégies d'allocation de ressources ont été proposées dont l'objectif est de minimiser deux des critères de qualité de service les plus importants, à savoir le temps et le coût d'exécution. De plus, les approches proposées tiennent en compte d'un troisième critère, contrairement aux approches proposées dans les chapitres précédents, pour assurer l'équité quant au partage des ressources disponibles. Par ailleurs, nous avons étudié deux scénarios en ce concerne le processus des arrivées des instances à exécuter dans le systèmes : (1) toutes les instances arrivent en même temps et (2) les instances arrivent au fur et à mesure.

Les résultats numériques obtenus sont encourageants. En effet, les approches proposées exhibent des résultats satisfaisants dans le sens de l'assurance de l'équité de partage des ressources. Cependant, il convient de noter qu'il n'est pas facile de choisir une des stratégies proposées au détriment des autres sauf dans certains cas particuliers discutés précédemment. C'est pour cela d'ailleurs, que nous recommandons aux utilisateurs d'utiliser toutes les stratégies simultanément.

Chapitre 8

Conclusion générale et perspectives

Le *Cloud Computing* a complètement changé la façon dont les ressources informatiques sont consommées via le Web en introduisant un nouveau modèle économique basé sur « payer en fonction de la consommation réelle ». Ce modèle s'oppose au modèle traditionnel d'acquisition de licences logiciels déployés sur les serveurs de l'entreprise utilisatrice, qui demande de nombreuses opérations telles que les mises à jour, la configuration, les tests et l'exécution, engendrant ainsi d'importants coûts. Dans la perspective de réduire ces coûts et ce type de difficultés, un nouveau paradigme a vu le jour. Il s'agit du *Cloud Computing*. Il permet, en effet, l'utilisation des ressources de calcul et/ou de stockage de données sans se soucier de la gestion de l'infrastructure sous-jacente. Cette gestion est du ressort du fournisseur des services en question. A cause de ces nombreux avantages avérés, les entreprises n'ont cessé d'y migrer leurs applications.

Cependant, tandis qu'un engouement majeur et incontestable soutient le développement du *Cloud*, des doutes liés aux questions sécuritaires et de qualité de service peuvent freiner son essor et même compromettre son avenir. L'un de ces problèmes fondamentaux, faisant partie de la gestion optimale des ressources disponibles, est l'ordonnancement des processus. Ce dernier consiste à déterminer l'ordre dans lequel les différentes tâches des processus doivent s'exécuter et les ressources que chacune d'elles doit utiliser. En outre, les utilisateurs sont confrontés au problème du choix des meilleures ressources (fournisseurs) à utiliser.

Cette dernière constitue la principale raison qui régit ce travail. Plus précisément, nous nous sommes intéressés au problème du choix des meilleurs ressources (fournisseurs) à utiliser de façon à satisfaire au mieux leurs utilisateurs. Le problème d'allocation de ressources et d'ordonnancement des graphes de tâches a été largement étudié dans le do-

maine des grilles de calcul en présence notamment de ressources hétérogènes. Mais il reste ouvert dans le domaine du *Cloud Computing*. En outre, il n'existe aucun algorithme d'ordonnancement lorsque plusieurs instances d'un processus accèdent de manière concurrente aux ressources disponibles.

Nous donnons dans ce chapitre un résumé du contenu du rapport, en faisant ressortir les perspectives possibles. Le but de ce travail est de développer des méthodes d'allocation de ressources et d'ordonnancement de processus dans le cadre du *Cloud Computing* en prenant en compte des critères conflictuels d'optimisation, à savoir le temps d'exécution, le coût engendré par l'utilisation d'un ensemble de ressources ainsi que le critère d'équité lorsque plusieurs instances du même processus accèdent aux ressources disponibles de manière concurrente. D'autre part, les approches proposées prennent en compte le fait que plusieurs fournisseurs (ressources) peuvent accomplir les mêmes tâches.

8.1 Synthèse des travaux

Dans cette thèse, nous avons abordé le problème d'allocation de ressources et d'ordonnancement de tâches de processus scientifiques et métiers sous trois aspects complémentaires que nous détaillons par la suite :

1. Le premier aspect consiste à proposer des méthodes d'allocation de ressources et d'ordonnancement de tâche d'un processus scientifique dont toutes les tâches sont supposées être automatisées.

2. Le deuxième aspect consiste à proposer des méthodes d'allocation de ressources et d'ordonnancement de tâche d'un processus dont un sous-ensemble de tâches le composant nécessitent l'intervention d'une ressource humaine pour les accomplir. Pour cela, nous avons proposé d'utiliser des modèles de prévision pour prédire la disponibilité des ressources utilisées.

3. Le troisième aspect est une généralisation des deux premières approches dans le sens où nous nous sommes intéressés au problème d'allocation de ressource et d'ordonnancement de tâches d'un ensemble d'instances d'un processus métier donné.

Proposition d'approches d'allocation de ressources et d'ordonnancement des processus scientifiques

Dans un premier temps, nous avons proposé un modèle pour l'allocation des ressources et l'ordonnancement des tâches d'un processus scientifique. Le cadre de l'analyse est celui des graphes de tâches sans cycle et la technique utilisée est celle de l'optimisation bi-objectifs. Concrètement, nous avons proposé trois approches complémentaires dont les deux premières sont respectivement basées sur la fonction objectif temps d'exécution et la fonction objectif coût d'exécution. Quant à la troisième approche, elle est basée sur les deux premières approches en ne sélectionnant au final que les solutions dites non-dominées (Pareto). Notre point de départ est basé sur les approches proposées pour l'ordonnancement des tâches des processus scientifiques dans le cadre des grilles de calcul. Cependant, comme mentionné auparavant, ces approches présentent de nombreuses insuffisances dont (1) le nombre de ressources supposé limité et (2) la non prise en compte de la dimension économique du *Cloud*. Dans ce sens, nous avons proposé des approches plus adaptées dans le cas de l'utilisation de ressources hétérogènes déployées dans le cadre de ce paradigme. Afin de prendre en compte son élasticité, nous avons supposé que le nombre de ressources mises à la disposition des utilisateurs sont en nombre infini. D'autre part, sa dimension économique est prise en compte dans le modèle que l'on propose en visant à optimiser la fonction objectif coût engendrée par l'utilisation de ses ressources pour l'exécution d'un processus scientifique. Dans cette première contribution, toutes les tâches sont supposées automatisées. Par conséquent, nous n'avons considéré que les ressources informatiques (c'es-à-dire les machines virtuelles). D'après les résultats numériques obtenus aucun critère d'optimisation pris en compte ne domine l'autre. De plus, aucune stratégie de parcours du graphe modélisant le processus en question n'est meilleure que l'autre. C'est pour cela, que nous recommandons l'utilisation de toutes les stratégies simultanément.

Proposition d'approches d'allocation de ressources et d'ordonnancement des processus métiers

Il est évident que les approches proposées précédemment ne sont pas adaptées lorsque l'on exécute un processus métier. En effet, contrairement à un processus scientifique, où toutes les tâches sont généralement automatisées, dans le cas d'un processus métier, il est souvent difficile voire impossible d'automatiser toutes les tâches. Pour cela, nous avons

distinguer deux types de ressources : (1) ressources humaines pour exécuter les tâches non automatisées et (2) machine virtuelles pour exécuter les tâches automatisées. L'introduction de ressources humaines, qui sont bien en nombre fini, engendrent deux difficultés supplémentaires. La première est celle du partage d'une même ressource par plusieurs tâches. La deuxième difficulté est celle de la gestion de l'*opacité* de travail des ressources humaines. Autrement dit, d'autres processus peuvent y avoir accès. Afin d'y remédier, nous avons proposé d'utiliser des modèles de prévision pour estimer les disponibilités de ces ressources. Les contributions dans cette deuxième partie suivent le même schéma que la contribution précédente. C'est-à-dire, trois approches ont été proposées basées respectivement sur le temps, le coût d'exécution et les deux fonctions simultanément.

Prise en compte de l'exécution de plusieurs instances simultanément

Il est rare lorsque l'on exécute un processus métier de n'en exécuter qu'une seule instance. En effet, généralement plusieurs instances sont exécutées simultanément. Par conséquent, il convient d'assurer une certaine équité quant à l'accès aux ressources partagées. L'équité que nous avons introduit fait référence au fait que lorsque les ressources sont partagées le temps d'exécution, de chacune des instances qui y accèdent, est supérieur à son temps d'exécution lorsque toutes les ressources sont à sa disposition et à elle seule. De plus, comme dans les deux cas précédents nous avons distingués deux types de ressources : humaines et machines virtuelles. Nous avons proposé un ensemble de stratégies d'allocation des ressources disponibles en prenant toujours en compte les deux critères de qualité de service : le temps et le coût d'exécution.

8.2 Perspectives

Suite aux travaux présentés dans cette thèse, nous envisageons de nombreuses perspectives dont les plus significatives sont résumés ci-dessous.

Prise en compte d'autres critères de qualité de service et des phénomènes aléatoires sous-jacents à l'exécution des processus

Nous travaillons actuellement sur la prise en compte dans la sélection des meilleures ressources à utiliser d'autres critères de qualité de service tels que la sécurité, la disponibilité dans le sens de la probabilité qu'une ressource soit disponible pour exécuter une

tâche donnée. De plus, il serait intéressant de prendre en compte le fait par exemple que le temps d'exécution d'une ressource n'est pas une constante mais une variable aléatoire suivant une loi de probabilité donnée. Pour cela, nous envisageons d'adapter nos approches à ce dernier cas en utilisant des techniques de l'optimisation stochastique.

Introduction d'une nouvelle notion d'optimalité

La notion d'optimalité de Pareto que nous avons retenue pour sélectionner les solutions générées par nos approches a une insuffisance majeure. En effet, elle permet de sélectionner par exemple deux solutions dont l'écart sur un des critères est très significatif pour un utilisateur donné. Concrètement, si l'on compare deux solutions (ordonnancement) sur trois critères à minimiser, les deux solutions suivantes seront sélectionnées $(12, 41, 8)$ et $(10, 99, 6)$. Or, la différence entre les valeurs du deuxième critère (41 et 99) est significative et pourrait l'être d'avantage si on doit la multiplier par une constance (exemple : million d'euros). Afin de pallier cette insuffisante, nous travaillons actuellement sur l'introduction d'une nouvelle notion d'optimalité définie par deux indices :

1. Indice de concordance (*prise en compte de la majorité*) :
2. Indice de non discordance :

L'objectif essentiel de l'introduction de ces deux indices est « la démocratisation du processus de décision » dans le sens où une hypothèse de surclassement (un ordonnancement o_i est meilleur que l'ordonnancement o_j) sera retenue ou remise en cause à l'issue du test de non-discordance. Une idée intéressante serait d'établir la relation entre la notion d'optimalité au sens de Pareto et celle que nous venons d'introduire et de proposer ainsi des algorithmes efficaces pour la sélection des solutions vérifiant les deux tests introduits ci-dessus.

Extension des approches proposées au cas de tâches modulables

Dans les applications parallèles, généralement, deux types de parallélisme peuvent être distingués, à savoir : (1) le parallélisme des tâches et (2) le parallélisme des données. Dans les travaux présentés dans cette thèse ces deux types de parallélisme sont intrinsèquement liés. Autrement dit, les tâches et les données sont respectivement exécutées et sauvegardées sur une même ressource (machine virtuelle). Une idée intéressante serait de séparer ces deux parallélismes. Par ailleurs, dans nos travaux, les tâches constituant un processus donné sont supposées indivisibles dans le sens où une tâche est exécutée par une et une

seule machine. Afin de tirer profit au maximum du paradigme *Cloud Computing*, nous travaillons actuellement sur l'extension des travaux présentés dans cette thèse pour prendre en compte le fait qu'une tâche donnée peut être divisée et par conséquent les parties ainsi résultantes peuvent être exécutées par des ressources distinctes.

Extension des approches proposées à un réseau de processus

Nous envisageons également d'étendre les approches proposées à un réseau de processus constituant une organisation donnée et qui échangent des flux d'informations, de matières, ou des flux financiers. Cet écosystème de multiples modèles de processus vise à créer de la valeur pour ladite organisation. La performance collective doit être analysée, mesurée et si besoin améliorée.

Glossaire

DAG : Directed Acyclic Graph (graphe orienté sans cycle).

$G = (T, E)$: graphe orienté sans cycle.

T : ensemble des tâches d'un processus.

E : ensemble des dépendances entre les tâches.

t_{input} : tâche initial d'un processus (tâches sans prédécesseurs immédiats).

t_{exit} : tâche représentant la fin d'un processus (tâches sans successeurs immédiats).

(t_i, t_j) : arc représentant une contrainte de précédence entre la tâche t_i et la tâche t_j.

$data[i, j]$: données transmises de la ressource qui exécute t_i à la ressource qui exécute t_j.

RG : graphe de ressources.

$VM = \{VM_1, ..., VM_m\}$: ensemble des catégories de machines virtuelles.

(VM_i, VM_j) : représente le lien entre les deux machines virtuelles VM_i et VM_j.

B : matrice des débit entre les différentes machines virtuelles.

$B[i, j]$: le débit entre la machine virtuelle VM_i et la machine virtuelle VM_j.

$TT(VM(t_i), VM(t_j))$: le temps de transfert entre les deux machines virtuelle $VM(t_i)$ et $VM(t_j)$.

$ET(t_j, VM_i)$: exécution de la tâche t_j par la machine virtuelle VM_i

$ET(t_j, r_i)$: exécution de la tâche t_j par la ressource r_i

$UEC(VM_j)$: le coût engendré par l'utilisation de la machine virtuelle VM_j.

$EC(t, VM_j)$: le coût d'exécution engendré par l'utilisation de la machine virtuelle VM_j.

$\mathrm{TC}(VM(t_i), VM(t_j))$: le coût de transfert entres les deux machines virtuelles $VM(t_i)$ et $VM(t_j)$.

$C_{out}(VM(t_i))$: le coût dû au transfert des données générées par l'exaction de la tâche t_i sur la machine virtuelle $VM(t_i)$.

$C_{in}(VM(t_i))$: le coût dû à la réception des données nécessaires pour exécuter la tâche t_i.

$ST(t_j)$: la date de début ai plus tôt de l'exécution de la tâche t_j.

$FT(t_j)$: la date de fin au plus tôt de l'exécution de la tâche t_j

$pred(t_j)$: ensemble des prédécesseurs immédiats de la tâche t_j.

$succ(t_j)$: ensemble des successeurs immédiats de la tâche t_j.

makespan : la fonction objectif du temps global d'exécution.

cost : la fonction global du coût d'exécution.

L : le nombre des niveau du graphe représentant un processus.

LB : borne inférieure.

V_{algol} : valeur du temps ou du coût obtenue par nos algorithmes.

V_{LB} : valeur de de la borne inférieure.

$\delta_{LB-algo}$: écart entre la valeur de obtenue par nos algorithmes et la valeur de la borne inférieure.

$HR = \{HR_1, ..., HR_m\}$: ensemble des ressources humaines.

$R(t_i)$: la ressource humaine qui exécute ma tâche t_i.

$Avail[r_j, t_i]$: date à laquelle la ressource r_j est prête à commencer l'exécution de la tâche t_i.

slowdown : facteur de ralentissement.

$slowdown_{time}(I_j)$: valeur du facteur de ralentissement de l'instance I_j considérant le temps global d'exécution.

$cost(I_j)$: le coût global d'exécution de l'instance I_j.

$ccost(I_j)$: le rapport entre le coût d'exécution de l'instance I_j lorsque elle dispose à elle seule de toutes les ressources et le coût de son exécution lorsque les ressources diposinibles sont partagées.

$FT_s^{I_j}$ est le temps global d'exécution de l'instance I_j quand toutes les ressources sont mises à disposition et à elle seule. Autrement dit, l'ensemble des instances à exécuter est réduit à l'instance I_j.

$FT_m^{I_j}(t_{exit})$ est le temps global d'exécution de l'instance I_j lorsque les ressources utilisées sont éventuellement partagées par d'autres instances de l'ensemble I.

$cost_s^{I_j}$ est le coût global d'exécution de l'instance I_j quand toutes les ressources sont mises à sa disposition et à elle seule. C'est-à-dire l'ensemble $I = \{I_j\}$.

$cost_m^{I_j}$ est le coût global d'exécution de l'instance I_j lorsque les ressources utilisées pour l'exécuter sont éventuellement utilisées par d'autres instances de l'ensemble I.

$slowdown(I_j)$ est le facteur de ralentissement de l'instance I_j.

$slowdown(t_i)$ est le facteur de ralentissement de la tâche t_i.

l_k est l'ensemble des tâches du niveau k du graphe modélisant un processus.

$age(I_j)$ est le temps qui s'est écoulé depuis l'arrivée de l'instance I_j dans le système.

Bibliographie

[1] N. Capit, G. Da Costa, Y. Georgiou, G. Huard, C. Martin, G. Mounié, P. Neyron, and O. Richard, **A Batch Scheduler With High Level Components**. In Proceedings of the 5th International Symposium on Cluster Computing and the Grid (CCGrid 2005), Cardiff, UK, May 2005.

[2] OAR, **Resource Management System for High Performance Computing**, http://oar.imag.fr.

[3] R. L. Henderson, **Job Scheduling Under the Portable Batch System**. In Processing of the Job Scheduling Strategies for Parallel Processing (JSSPP 1995), pages 279-294, Santa Barbara, CA, USA, April 1995.

[4] D. B. Jackson, Q. Snell, and M. J. Clement, **Core Algorithms of the Maui Scheduler**, In Proceedings of the 7th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2001), pages 87-102, Cambridge, MA, USA, June 2001.

[5] A. W. Mu'alem and D. G. Feitelson, **Utilization, Predictability, Workloads, and User Runtime Estimates in Scheduling the IBM SP2 with Backfilling**. IEEE Transactions on Parallel and Distributed Systems, 12(6) : 529-543, 2001.

[6] W. Smith, V. E. Taylor, and I. T. Foster, **Using Run-Time Predictions to Estimate Queue Wait Times and Improve Scheduler Performance**, In Proceedings of the Job Scheduling Strategies for Parallel Processing (JSSPP'99), pages 202-219, San Juan, Puerto Rico, April 1999. Springer-Verlag.

[7] D. A. Lifka, **The ANL/IBM SP Scheduling System**, In Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP'95), pages 295-303, Santa Barbara, CA, USA, April 1995. Springer-Verlag.

[8] S. H. Bokhari, **On the Mapping Problem**, IEEE Transactions on Computers, 30(3) : 207-214, 1981.

[9] C.-C. Hui and S. T. Chanson, **Allocating Task Interaction Graphs to Processors in Heterogeneous Networks**, IEEE Transactions on Parallel and Distributed Systems, 08(9) : 908-925, 1997.

[10] M. R. Garey and D. S. Johnson, **Computers and Intractability : A Guide to the Theory of NP-Completeness**, W. H. Freeman, January 1979.

[11] H. Topcuouglu, S. Hariri, and M.-Y. Wu, **Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing**, IEEE Transactions on Parallel and Distributed Systems, 13(3) : 260-274, 2002.

[12] G. C. Sih and E. A. Lee, **A Compile-Time Scheduling Heuristic for Interconnection Constrained Heterogeneous Processor Architectures**, IEEE Transactions on Parallel and Distributed Systems, 04(2) : 175-187, 1993

[13] M.Y. Wu and D. D. Gajski., **Hypertool : A Programming Aid for Message-Passing Systems**, IEEE Transactions on Parallel and Distributed Systems, 01(3) : 330-343, 1990.

[14] T. Yang and A. Gerasoulis, **DSC : Scheduling Parallel Tasks on an Unbounded Number of Processors**, IEEE Transactions on Parallel and Distributed Systems, 5(9) : 951-967, 1994.

[15] M. A. Palis, J.-C. Liou, and D. S. L. Wei, **Task Clustering and Scheduling for Distributed Memory Parallel Architectures**, IEEE Transactions on Parallel Distributed Systems, 7(1) : 46-55, 1996.

[16] G. L. Park, B. Shirazi, and J. Marquis, **DFRN : A New Approach for Duplication Based Scheduling for Distributed Memory Multiprocessor Systems**, In Proceedings of the 11th International Symposium on Parallel Processing(IPPS'97), pages 157-166, Geneva, Switzerland, April 1997. IEEE Computer Society.

[17] Y. C. Chung and S. Ranka, **Applications and Performance Analysis of a Compile- time Optimization Approach for List Scheduling Algorithms on Distributed Memory Multiprocessors**, In Proceedings of the 1992 ACM/IEEE conference on Supercomputing (Supercomputing'92), pages 512-521, Minneapolis, MN, USA, 1992. IEEE Computer Society Press.

[18] I. Ahmad and Y. Kwok, **A New Approach to Scheduling Parallel Programs Using Task Duplication**, In Proceedings of the 1994 International

Conference on Parallel Processing (ICPP' 94), pages 47-51, Washington, DC, USA, 1994. IEEE Computer Society.

[19] R. Bajaj and D. P. Agrawal, **Improving Scheduling of Tasks in a Heterogeneous Environment**, IEEE Transactions on Parallel and Distributed Systems, 15(2) : 107-118, 2004.

[20] N. Vydyanathan, S. Krishnamoorthy, G. Sabin, U. Catalyurek, T. Kurc, P. Sadayap, and J. Saltz, **An Integrated Approach for Processor Allocation and Scheduling of Mixed-Parallel Applications**, In Proceedings of the International Conference on Parallel Processing (ICPP'06), Columbus, Ohio, USA, August 2006.

[21] M. Skutella, **Approximation Algorithms for the Discrete Time-Cost Tradeoff Problem**, Mathematics of Operations Research, 23(4) : 909-929, 1998.

[22] A. Radulescu and A. van Gemund, **A Low-Cost Approach towards Mixed Task and Data Parallel Scheduling**, In Proceedings of the 15th International Conference on Parallel Processing (ICPP 2001), Valencia, Spain, September 2001.

[23] A. Radulescu, C. Nicolescu, A. J. C. van Gemund, and P. P. Jonker, **CPR : Mixed Task and Data Parallel Scheduling for Distributed Systems**, In Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS'01), San Francisco, CA, USA, April 2001. IEEE Computer Society.

[24] R. Lepère, G. Mounié, and D. Trystram, **An approximation algorithm for scheduling trees of malleable tasks**, European Journal of Operational Research, 142(2) : 242-249, 2002.

[25] S. Bansal, P. Kumar, and K. Singh, **An Improved Two-step Algorithm for Task and Data Parallel Scheduling in Distributed Memory Machines**, Parallel Computing, 32(10) : 759-774, 2006.

[26] P. Shroff, D. W. Watson, N. Flann, and R. Freund, **Genetic Simulated Annealing for Scheduling Datadependent Tasks in Heterogeneous Environments**, In Proceedings of the 5th IEEE Heterogeneous Computing Workshop (HCW'96), pages 98-117, Honolulu, Hawaii, April 1996.

[27] H. J. Siegel, V. P. Roychowdhury, and L. Wang, **A Genetic-Algorithm-Based Approach for Task Matching and Scheduling in Heterogeneous Computing Environment**, In Proceedings of the 5th IEEE Heterogeneous Computing Workshop (HCW'96), pages 72-85, Honolulu, Hawaii, April 1996.

[28] H. Singh and A. Youssef, **Mapping and Scheduling Heterogeneous Task Graphs Using Genetic Algorithms**, In Proceedings of the 5th IEEE Heterogeneous Computing Workshop (HCW'96), pages 86-97, Honolulu, Hawaii, April 1996.

[29] G. Sabin, G. Kochhar, and P. Sadayappan, **Job Fairness in Non-Preemptive Job Scheduling**, In Proceedings of the 2004 International Conference on Parallel Processing (ICPP'04), pages 186-194, Washington, DC, USA, 2004. IEEE Computer Society.

[30] U. Schwiegelshohn and R. Yahyapour, **Fairness in Parallel job scheduling**, Journal of Scheduling, 3(5) : 297-320, 2000.

[31] D. B. Jackson, Q. Snell, and M. J. Clement, **Core Algorithms of the Maui Scheduler**, In Proceedings of the 7th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2001), pages 87-102, Cambridge, MA, USA, June 2001.

[32] H. Zhao and R. Sakellariou, **Scheduling Multiple DAGs onto Heterogeneous Systems**, In Proceedings of the 15th Heterogeneous Computing Workshop (HCW'06), Island of Rhodes, Greece, April 2006.

[33] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, **Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems**, In Proceedings of the Eighth Heterogeneous Computing Workshop (HCW'99), pages 30-46, San Juan, Puerto Rico, April 1999. IEEE Computer Society.

[34] Beaumont, O., Legrand, A., Marchal, L., Robert, Y, **Steady-state scheduling on heterogeneous clusters : why and how ?** In : 6th Workshop on Advances in Parallel and Distributed Computational Models (APDCM 2004). IEEE Computer Society Press, Santa Fe, USA.

[35] O. Beaumont, A. Legrand, L. Marchal and Y. Robert, **Assessing the impact and limits of steady-state scheduling for mixed task and data parallelism on heterogeneous platforms**, In Proceedings of the Third International

Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks, pages 296-302, ISPDC'04, Cork, Ireland, IEEE Computer Society.

[36] S. Diakité, L. Marchal, J.M. Nicod, and L. Philippe, **Steady-state for batches of identical task trees**, In Proceedings of the 15th International Euro-Par Conference on Parallel Processing, pages 203-215, Euro-Par'09, Delft, The Netherlands, Springer-Verlag, Berlin, Heidelberg.

[37] D. G. Feitelson and L. Rudolph, **Towards Convergence in Job Schedulers for Parallel Supercomputers**, In Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 1996), pages 1-26, Honolulu, Hawaii, April 1996. Springer-Verlag.

[38] G. M. Amdahl, **Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilitie**, In Proceedings of the AFIPS 1967 Spring Joint Computer Conference, volume 30, pages 483-485, April 1967.

[39] I. Foster and C. Kesselman, **Globus : A metacomputing infrastructure toolkit**, International Journal of Supercomputer Applications, pages 115-128, 1997.

[40] M. Weske, **Business Process Management : Concepts, Languages, Architectures**, Springer Verlag, first edition, November 2007.

[41] P. A. Buhler and J. M. Vidal, **Towards adaptive workflow enactment using multiagent systems**, Information Technology and Management, 6(1) : 61-87, jan 2005.

[42] R. F. Brena, J. L. Aguirre, C. I. Chesñevar, E. Ramírez, and L. Garrido, **Knowledge and information distribution leveraged by intel- ligent agents**, Knowledge and Information Systems (KAIS), Springer Verlag, 2007.

[43] M. Sonntag and D. Karastoyanova, **Next Generation Interactive Scientific Experimenting Based On The Workflow Technology**, 21st IASTED International Conference on Modelling and Simulation, Banff, Canada, July 2010.

[44] D. Akram et al., **Evaluation of BPEL to scientific workflows**, 6th IEEE International Symposium on Cluster Computing and the Grid, 2006.

[45] B. Wassermann et al., **Sedna : A BPEL-based environment for scientific**

workflow modeling, Workflows for e-Science and Scientific Workflows for Grids, I. Taylor et al., Eds. Springer, 2007.

[46] I. Wassink et al., **Designing workflows on the fly using e-BioFlow**, Int Conf. on Service Oriented Computing, Stockholm, Sweden, 2009.

[47] R. Barga and D.B. Gannon, **Scientific versus business workflows**, in Workflows for e-Science : Scientific Workflows for Grids, I. Taylor, E. Deelman, D.B. Gannon, M. Shields, Eds. Springer, 2007.

[48] P. Vassiliadis and A. Simitsis, **Extraction, Transformation, and Loading**, In Encyclopedia of Database Systems, pages 1095-1101. 2009.

[49] A. Pointet, R. Caloz and M. Riedo, **Compression des images de télédétection**, 2002.

[50] U. Yildiz, A. Guabtni and A. H.H. Ngu, **Business versus Scientific Workflows : A Comparative Study Services**, IEEE Congress, vol. 1, pages 340-343, 2009.

[51] B. Adam and J. Van Hemert, **Scientific workflow : a survey and research directions**, In Proceedings of the 7th international conference on Parallel processing and applied mathematics, PPAM'07, pages 746-753, Berlin, Heidelberg, 2008. Springer-Verlag.

[52] E. Deelman, D. Gannon, M. Shields and I. Taylor, **Workflows and e-Science : An overview of workflow system features and capabilities. Future Generation Computer Systems**, vol. 25, no. 5, pages 528-540, 2009.

[53] T. Crusson, **Bpm : de la modélisation à l'exécution positionnement par rapport aux architectures orientées services**, white paper, 2003.

[54] **Object management group**, http://www.omg.org/

[55] M. Mazzucco, D. Dyachuk and R. Deters, **Maximizing Cloud Providers' Revenues via Energy Aware Allocation Policies**, IEEE 3rd International Conference on Cloud Computing (CLOUD),131-138, 2010.

[56] W. Kim, **Cloud Computing : Today and Tomorrow**, ETH Zurich, Chair of Software Engineering Vol. 8, No. 1, 2009.

[57] F. Armando, G. R. Joseph, A. Konwinski, G. Lee, D. Patterson, A. Rabkin and I. Stoica, **Above the clouds : A Berkeley view of cloud computing**, Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS,28, 2009.

[58] P. Mell, T. Grance, **The NIST Definition of Cloud Computing**, (Draft)-Recommendations of the National Institute of Standards and Technology, Special publication 800-145 (draft), Gaithersburg (MD).

[59] E.Goldberg, **Genetic Algorithm in Search Optimization, and Machine Learning**, Addison-wesley, Reading, Massachusetts, 1989.

[60] D. Schaffer, **Multiple Objective Optimization with Vector Evaluated Genetic Algorithm**, In Genetic Algorithm ad their Applications : Proceedings of the first International conference on Genetic Algorithm, pages 93-100, 1985.

[61] V. Pareto, **Cours d'économie politique**, vol. 1 et 2., F. Rouge, Lausanne, 1896.

[62] G. Juve and E. Deelman, **Scientific Workflows in the Cloud, in Grids, Clouds and Virtualization**, M. Cafaro Eds. Springer, pages 71-91, 2010.

[63] Z. Li, Y. Bai, H. Zhang and Y. Ma, **Affinity-Aware Dynamic Pinning Scheduling for Virtual Machines in Cloud Computing Technology and Science**, (CloudCom2010) , pages 242-249, 2010.

[64] T.T. Huu and J. Montagnat, **Virtual Resources Allocation for Workflow-Based Applications Distribution on a Cloud Infrastructure**, 10th IEEE/ACM International Conference Cluster, Cloud and Grid Computing (CCGrid), pages 612-617, 2010.

[65] J. Tordsson, R. Montero, R. Moreno-Vozmediano, and I. Llorente,**Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers**, Future Generation Computer Systems, Volume 28, Issue 2, pages 358-367, 2012.

[66] K. Bessai, S.Youcef, Ammar Oulamara, Claude Godart S.Nurcan, **Business Process scheduling strategies in Cloud environments with fairness metrics**. IEEE SCC, 10th International Conference on Services Computing June 27-July 2, 2013 Santa Clara Marriott, CA, USA (Center of Silicon Valley).

[67] K. Bessai, S. Youcef, A. Oulamara, C. Godart, **Bi-criteria strategies for business processes scheduling in cloud environments with fairness metrics**. RCIS 2013 : 1-10.

[68] K. Bessai, S. Youcef, A. Oulamara, C. Godart, **Resources allocation and scheduling approaches for business process applications in Cloud**

contexts. A paraître dans le journal **International Journal of Grid and High Performance Computing (2013)**.

[69] K. Bessai, S. Youcef, A. Oulamara, C. Godart, S. Nurcan. **Resources allocation and scheduling approaches for business process applications in Cloud contexts**. 4th IEEE International Conference on Cloud Computing technology and Science (Cloud Com'2012), Taipei, Taiwan, December 3-6, 2012.

[70] K. Bessai, S. Youcef, C. Godart, S. Nurcan. **Business Process Compositions : Preserving k-soundness property**. The 2012 IEEE Asia-Pacific Services Computing Conference (APSCC'2012), Guilin, China, December 6-8, 2012.

[71] K. Bessai, S. Youcef, C. Godart, A. Oulamara, S. Nurcan. **Multi-objective resources allocation approaches for workflow applications in Cloud environments** (Poster). 20th International Conference on Cooperative Information Systems (CoopIS'2012), Roma, Italy, September 10-14, 2012.

[72] K. Bessai, S. Nurcan. **An actor-driven approach for business processes - How to take into account the environment of work ?** The 10th working Conference on Business Process Modelling, Development, and Support (BPMDS'09) (in association with the CAISE'09 Conference), June 8-9, 2009, Amsterdam, Netherlands.

[73] K. Bessai, B. Claudepierre, O. Saidani, S. Nurcan. **Context-aware Business Process Evaluation and Redesign**. The 9th Working Conference on Business Process Modelling, Development, and Support (BPMDS'08, in association with the CAISE'08 Conference), June 16-17, 2008, Montpellier, France.

[74] K. Bessai, S. Nurcan. **Composition de Workflow-nets : condition de préservation de la k-soundness**, IESI, Ingénierie d'Entreprise et de Systèmes d'Information INFORSID'2010 Marseille, France.

[75] K. Bessai **Gestion de la variabilité dans la modélisation et l'exécution des processus** ; 27eme congrès d'Inforsid , 26-27 Mai 2009, forum jeune chercheur, Toulouse, France.

[76] K. Bessai, S. Nurcan. **Guider le choix d'un formalisme de modélisation de processus : Démarche multicritère basée sur les patrons**. 26ème congrès INFORSID, 27-30 Mai 2008, Fontainebleau, France.

[77] H. Chen and M. Maheswaran. **Distributed Dynamic Scheduling of Composite Tasks on Grid Systems**, In Proceedings of the 12th Heterogeneous Computing Workshop (HCW'02), Fort Lauderdale, FL, April 2002.

[78] M. A. Iverson and F. Ozguner. **Hierarchical, Competitive Scheduling of Multiple DAGs in a Dynamic Heterogeneous Environment.** Distributed System Engeneering, 6(3) : 112-124, 1999.

[79] M. Xu, L. Cui, H. Wang and Y. Bi, **A Multiple QoS Constrained Scheduling Strategy of Multiple Workflows for Cloud Computing**, Parallel and Distributed Processing with Applications, 2009 IEEE International Symposium on Cloud, pages 629-634.

[80] K. Yu-Kwong and A. Ishfaq, **Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors**, Volume 31 Issue 4, Dec. 1999 Pages 406-471.

[81] J. Yu, M. Kirley and R. Buyya, **Multi-objective planning for workflow execution on Grids**, In Proceedings of the 8th IEEE/ACM International Conference on Grid Computing (GRID'07), IEEE Computer Society, Washington, DC, USA, 2007.

[82] B. Roger and G. Dennis, **Scientific versus Business Workflows**, In Workflows for e-Science, editor Taylor, Ian J. and Deelman, Ewa and Gannon, DennisB. and Shields, Matthew Springer London 2007.

[83] X. Jiajie, L. Chengfei and Z. Xiaohui, **Resource Planning for Massive Number of Process Instances**, On the Move to Meaningful Internet Systems : OTM 2009, Springer Berlin Heidelberg, pages 219-236.

[84] L. Wu, S. Kumar Garg and R. Buyya, **SLA-Based Resource Allocation for Software as a Service Provider (SaaS) in Cloud Computing Environments**, In Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID'11). IEEE Computer Society, Washington, DC, USA, pages 195-204.

[85] Z. Huang, W.M.P. van der Aalst and X. Lu and H. Duan., **Reinforcement learning based resource allocation in business process management**, Data and Knowledge Engineering, 2011.

[86] N. Russell, A.Hofstede, D. Edmond and Van der. Aalst, **Workflow Data Patterns**, QUT Technical report, Queensland University of Technology, 2004.

[87] N.Russell, A.Hofstede, D. Edmond and Van der. Aalst, **Workflow Resource Patterns**, BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven. 2004.

[88] C. Alexander, S. Ishikawa and M. Silverstein, **A pattern Language** , Oxford University Press, New York, 1977.

[89] W.M.P. van der Aalst, A.P. Barros, A.H.M. ter Hofstede and B. Kiepuszewski, **Advanced Workflow Pattern**, 7th International Conference on Cooperative Information Systems (CoopIS'2000).

[90] W.M.P. van der Aalst, A.P. Barros, A.H.M. ter Hofstede and B. Kiepuszewski, **Workflow Patterns** , BETA Working Paper Series, WP 47, Eindhoven University of Technology, Eindhoven, 2000.

[91] W.M.P. Van der Aalst, M. Weske and G. Wirtz, **Advanced Topics in Workflow Management : Issues, Requirements, and Solutions**, 2003.

[92] S. Nurcan and F. Daoudi, **A benchmarking framework for methods to design flexible business processes** , Improvement and Practice Journal on Business Process Management, Devellopement and Support , 2007.

[93] A. Stephen, **Introduction to BPMN**, IBM corporation, 2004.

[94] S. Nurcan, M.-H. Edme, **Intention Driven Modeling for Flexible Workflow Application**, Special issue of the software Process : Improvement and Practice Journal on Business Process Management, Development and Support, 2005.

[95] G. Alonso, D. El Abbadi, and C. Mohan, **Fonctionality and Limitation of Current Workflow Management Systems**, IEEE Expert, 1997.

[96] C. Rolland and S. Nurcan, **Contributions of Workflow to quality requirements**, Knowledge and Process Management : the Journal of Corporate Transformation, 2000.

[97] B. Roy, **Méthodes multicritères d'aide à la décision**, Economica, Collection Gestion, 1985.

[98] S. Nurcan, A. Hicheur, **A Comparative State-of-the-Art for Flexible Workflow Modeling**, The Business Process Management Tools and Technologies Information Resources Management Association International Conference, May 15-18, 2005.

[99] A. Krid and Y. MAO, **Etude comparative et argumentée sur la modélisation organisationnelle et opérationnelle des processus d'entreprises**, Mémoire de DEA , 2005.

[100] N. Russell, Wil M.P van der Aalst, A. H. M. ter Hofstede, P. Wohed, **On the Suitability of UML 2.0 Activity Diagrams for Business Process Modelling**, Proceedings of the Third Asia-Pacific Conference on Conceptual Modelling (APCCM 2006), volume 53 of CRPIT, pages 95-104, Hobart, Australie , 2006.

[101] P. Wohed, W.M.P. van der Aalst, M. Dumas, A.H.M. ter Hofstede and N. Russell, **On the Suitability of BPMN for Business Process Modelling**, In Proceedings of the 4th International Conference on Business Process Management (BPM 2006), Septembre 2006.

[102] P. Wohed, E. Perjons, M. Dumas, and A. ter Hofstede, **Pattern-Based Analysis of EAI Languages : The Case of the Business Modeling Language**, In Proceedings of the 5th International Conference on Enterprise Information Systems (ICEIS), France 2003.

[103] N. Russell, W.M.P. van der Aalst, A.H.M. ter Hofstede, and D. Edmond, **Workflow Resource Patterns : Identification, Representation and Tool Support**, Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE'05), volume 3520 of Lecture Notes in Computer Science, pages 216-232, Berlin, 2005.

[104] P. Wohed, W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede, **Pattern-Based Analysis of BPEL4WS**, QUT Technical report,Queensland University of Technology , Brisbane 2002.

[105] W.M.P. van der Aalst and A.H.M. ter Hofstede, **YAWL : Yet Another Workflow Language**, Information Systems, 30(4) : 245-275, 2005.

[106] W.M.P. van der Aalst, L. Aldred, M. Dumas, and A.H.M. ter Hofstede, **Design and Implementation of the YAWL System**, Proceedings of the 16th International Conference on Advanced Information Systems Engineering (CAiSE'04), 2004.

[107] R.T. Burlton, **Business Process Management - Profiting from process**, SAMS Publishing, 2001.

[108] A. Kumar, W.M.P. van der Aalst, and M.W. Verbeek, **Dynamic Workflow Distribution in Workflow Management Systems : How to balance quality and performance ?**, Journal of Management Information Systems 18(3) : 157-193, 2002.

[109] M. Zur, **Evaluation of Workflow Management Systems Using Meta Models**, Proceedings of the 32nd Hawaii International Conference on System Sciences, 1999.

[110] M. Zur, **Resource modeling in workflow applications**, Proceedings of the 1999 Workflow Management Conference, pages 137-153, November 1999.

[111] S. Nurcan, A. Etien, R. Kaabi, I. Zoukar, and C. Rolland, **Strategy Driven Business Process Modelling Approach**, Special issue of the Business Process Management Journal on Goal-oriented business process modeling, Emerald, 11 : 6, 2005.

[112] M. Nyanchama, S. L. Osborn, **The role graph model and conflict of interest**, ACM Transaction on Information and System Security, pages 3-33, 1999.

[113] G. Regev, and A. Wegmann, **Regulation-Based View on Business Process and Supporting System Flexibility**, Proceedings of the CAiSE'05 Workshop, 2005.

[114] M. Rosemann, M. Zur Muehlen, **Evaluation of workflow Management systems - A Meta Model Approach**, Australian Journal of Information Systems, 1998.

[115] C. Rolland, N. Prakash, **On the Adequate Modeling of Business Process Families**, 8th Workshop on Business Process Modeling, Development, and Support (BPMDS'07) in conjunction with CAISE'07(2007).

[116] C. Rolland, N. Prakash, and A. Benjamen, **A Multi-Model View of Process Modelling**, Requirements Engineering Journal (REJ), pages 169 - 187, 1999.

[117] C. Rolland, P. Loucopoulos, V. Kavakli, and S. Nurcan, **Intention based modelling of organisational change : an esperience report**, Proceedings of the fourth CAISE IFIP, International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD 99), Heidelberg, Germany, june 14-15, 1999.

[118] N. Russell, W.M.P van der Aalst, A.H.M. ter Hofstede, and D. Edmond, **Workflow Resource Patterns : Identification, Representation and Tool Sup-**

port, Proceedings of the 17th Conference on Advanced Information Systems Engineering CAiSE'05, 2005.

[119] N. Russell, A.H.M ter Hofstede, W.M.P. van der Aalst W.M.P, and N. Mulyar, **Workflow Control-Flow Patterns**, BPM Center Report BPM-06-22, BPMcenter.org, 2006.

[120] N. Russell, A. Hofstede, D. Edmond, W..P. Van der Aalst, **Workflow Resource Patterns**, BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven, 2004.

[121] O. Saidani and S. Nurcan, **A role based approach for modeling flexible business processes**, The 7th Workshop on Business Process Modelling, Development, and Support, BPMDS'06, in association with the CAISE'06 Conference, Springer Verlag, Juin 5-6, 2006.

[122] O. Saidani and S. Nurcan, **Towards situational Business Process Modeling**, 20th International Conference on Advanced Information Systems Engineering, CAISE'08 Forum, June 16-20, 2008.

[123] R. Simon and M. E. Zurko, **Separation of duty in role-based environments**, Proceedings of the 10th computer Security Fondations Workshop, pages 183-194, 1997.

[124] A. Barker J. and Hemert, **Scientific Workflow : A Survey and Research Directions**, Parallel Processing and Applied Mathematics, pages 746-753, Springer Berlin Heidelberg, 2008.

[125] L. Bittencourt, and E.R. Madeira, **HCOC : a cost optimization algorithm for workflow scheduling in hybrid cloud**, Journal of Internet Services and Applications, vol. 2, n. 3, 2011.

[126] B. Rajkumar, and M. Manzur, **GridSim : A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing**, CONCURRENCY AND COMPUTATION Journal : PRACTICE AND EXPERIENCE (CCPE), (14 :13), pages 1175–1220, 2002.

[127] M.S. Raunak, and L.J. Osterweil, **Resource Management for Complex, Dynamic Environments**, Software Engineering, IEEE Transactions, (39 :3), pages 384,402, March 2013.

[128] A. Azaron, H. Katagiri, M. Sakawa, K. Kato, and A. Memariani, **A multi-**

objective resource allocation problem in PERT networks, European Journal of Operational Research, (172 : 3), pages 838-854, 2006.

[129] M. Harchol-balter, M. E. Crovella and C. D. Murta, **On Choosing a Task Assignment Policy for a Distributed Server System**, IEEE Journal of Parallel and Distributed Computing, pages 231–242, 1999.

[130] M. Hammoud and M. F. Sakr, **Locality-Aware Reduce Task Scheduling for MapReduce**, In Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science (CLOUDCOM '11), IEEE Computer Society, Washington, DC, USA, pages 570-57, 2011.

[131] D. Candeia, R. Araujo, R. Lopes, and F. Brasileiro, **Investigating Business-Driven Cloudburst Schedulers for E-Science Bag-of-Tasks Applications**, Cloud Computing Technology and Science (CloudCom), pages 343-350, 2010.

[132] S. Genaud, and J. Gossa, **Cost-Wait Trade-Offs in Client-Side Resource Provisioning with Elastic Clouds**, Cloud Computing (CLOUD2011), pages 1-8, 2011.

[133] Q. Haiyang, D. Medhi, and T. Trivedi, **A hierarchical model to evaluate quality of experience of online services hosted by cloud computing**, IFIP/IEEE International Symposium on Integrated Network Management (IM), pages 105-112, 2011.

[134] K. Adrian, I. Fuyuki, and H. Shinichi, **Towards network-aware service composition in the cloud**, In Proceedings of the 21st international conference on World Wide Web (WWW '12). ACM, New York, NY, USA, pages 959-968, 2012.

[135] R. Buyya , R. Ranjan and N. Rodrigo, **Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit : Challenges and Opportunities**, CoRR,abs/0907, pages 48-78, 2009.

[136] S. G. Kumar, C. Shin Yeo, A. Anandasivam and R. Buyya, **Environment-conscious scheduling of HPC applications on distributed Cloud-oriented data center**, Journal of Parallel and Distributed Computing (71 :6), pages 732 -749, 2011.

[137] E. Juhnke, T. Dornemann, D. Bock, and B. Freisleben,, **Multi-objective Scheduling of BPEL Workflows in Geographically Distributed**

Clouds, IEEE International Conference on Cloud Computing (CLOUD), pages 412-419, 2011.

[138] K. Boloor, R. Chirkova, Y. Viniotis, and T. Salo, **Dynamic Request Allocation and Scheduling for Context Aware Applications Subject to a Percentile Response Time SLA**, in a Distributed Cloud IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), pages 464-472, 2010.

[139] D. Niyato, A.V. Vasilakos, and K. Zhu, **Resource and Revenue Sharing with Coalition Formation of Cloud Providers : Game Theoretic Approach**, 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pages 215-224, 2011.

[140] A. Oprescu, T. Kielmann, **Bag-of-Tasks Scheduling under Budget Constraints**, IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), pages 351-359, 2010.

[141] F. Zhang, J. Cao, K. Hwang, and C. Wu, **Ordinal Optimized Scheduling of Scientific Workflows in Elastic Compute Clouds**, In Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science, 2011.

[142] R. Van Bossche, K. Vanmechelen, and J. Broeckhove, **Cost-Efficient Scheduling Heuristics for Deadline Constrained Workloads on Hybrid Clouds**, IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom), pages 320-327, 2011.

[143] H. Zhao, and R. Sakellariou, **An Experimental Investigation into the Rank Function of the Heterogeneous Earliest Finish Time Scheduling Algorithm**, Euro-Par 2003 Parallel Processing, Springer Berlin Heidelberg, pages 189-194, 2003.

[144] M. Sonntag, D. Karastoyanova, and E. Deelman, **Bridging the Gap between Business and Scientific Workflows : Humans in the Loop of Scientific Workflows**, IEEE Sixth International Conference on e-Science (e-Science) pages 106-213, 2010.

[145] L. Wang, H. J. Siegel, V. R. Roychowdhury, and A. Maciejewski, **Task matching and scheduling in heterogeneous computing environments using**

a genetic-algorithm-based approach, J. Parallel Distrib. Comput. (47 :1), 1997.

[146] M.E. Frincu, N.M. Villegas, D. Petcu, and H. A. Muller, and R. Rouvoy, **Self-Healing Distributed Scheduling Platform**, 11th IEEE/ACM International Symposium on, Cluster, Cloud and Grid Computing (CCGrid), pages 225-234, 2011.

[147] H. Zhao and R. Sakellariou, **Scheduling multiple DAGs onto heterogeneous systems**, 20th International, Parallel and Distributed Processing Symposium, IPDPS 2006.

[148] Y. K. Kwok and I. Ahmed, **Benchmarking and Comparison of the Task Graph Scheduling Algorithms**, In Journal of Parallel and Distributed Computing, pages 381-422, 1999.

[149] A. Paul and J. M. Vidal, **Towards adaptive workflow enactment using multiagent systems**, Information Technology and Management, 6(1) : pages 61-87, 2005.

[150] F. Ramón, et al., **Knowledge and information distribution leveraged by intelligent agents**. Knowledge and Information Systems (KAIS), Springer Verlag, 2007.

[151] Hardin, G. **The tragedy of the commons**, Science 162, pages 1243-248, 1968.

[152] M. Bender, S. Chakrabarti, and S. Muthukrishnan., **Flow and stretch metrics for scheduling continuos job streams**, In Proc. of the 9th Annual ACM- SIAM Symposium on Discrete Algorithms, 1998.

[153] A. Silberschatz and P. Galvin, **Operating System Concepts**, 5th Edition. John Wiley & Sons, 1998.

[154] W. Stallings, **Operating Systems, 2nd Edition**, Prentice Hall, 1995.

[155] Uddi spec technical committee. **universal description, discovery and integration (uddi), draft version 3.0.2**., web site available at www.uddi.org/.

[156] **W3c. soap version 1.2, w3c specification**, web site available at www.w3.org/tr/soap/.

[157] **Web services description language (wsdl) 1.1, w3c recommendation**, web site available at www.w3.org/tr/wsdl.

[158] S. Al Aloussi, **SLA Business Management Based on Key Performance Indicators ?**, Proceedings of World Congress on Engineering 2012 Vol. 3, WCE 2012, July 4-6, 2012, London, U.K.

[159] M. Maurer, I. Brandic, R. Sakellariou **Enacting SLAs in Clouds Using Rules**, Euro-Par'11 Proceedings of the 17th international conference on Parallel processing, pages 455-466, Springer- Verlag Berlin, Heidelberg.

[160] F. Faniyi, R. Bahsoon, **Self-Managing SLA Compliance in Cloud Architectures : A Market-based Approach**, ISARC'12, june 26-28, Bertinoro, Italy, 2012.

[161] R. Buyya, S. K. Garg, and R. N. Calheiros, **SLA Oriented Resource Provisioning for Cloud Computing : Challenges, Architecture, and Solutions ?**, International Conference on Cloud and Service Computing, 2011.

[162] P. Mell, T. Grance, **The NIST Definition of Cloud Computing**, (Draft)-Recommendations of the National Institute of Standards and Technology. Special publication 800-145 (draft), Gaithersburg (MD).

[163] R. Buyya, S. Pandey and C. Vecchiola, **Cloudbus toolkit for market-oriented cloud computing**, In CloudCom'09 : Proceedings of the 1st International Conference on Cloud Computing, volume 5931 of LNCS, pages 24-44, Springer, Germany, December 2009.

[164] G. Juve, E. Deelman, K. Vahi,, G. Mehta, B. Berriman, B. P. Ber-man and P. Maechling, **Scientific workflow applications on Amazon EC2**. In Cloud Computing Workshop in Conjunction with e-Science, Oxford, UK, 2009, IEEE.

[165] A. V. Mladen, **Cloud Computing - Issues, Research and Implementations**, in Journal of Computing and Information Technology, (16 :4), pages 235-246, 2008.

[166] E. Deelman, D. Gannon, M. Shields and I. Taylor, **Workflows and e-Science : An overview of workflow system features and capabilities**, Future Generation Computer Systems, (25 :5), pages 528-540, 2009.

[167] H. Topcuoglu, S. Hariri, M. Y . Wu, **Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing**, IEEE Transactions on Parallel and Distributed Systems, (13 :3), 2002.

[168] J. D. Ullman, **NP-Complete Scheduling Problems**, J. Computer and Systems Sciences, Volume 10, pages 3 84-393, 1975.

[169] M. R. Gary and D. S. Johnson, **Computers and Intractability : A Guide to the Theory of NP-Completeness**, W.H. Freeman and Co., 1979.

[170] H. Hoogeveen,**Multicriteria scheduling**, In European Journal of Operational Research 167, pages 592-623, 2005.

[171] M. Wieczorek, S. Podlipnig, R. Prodan, T. Fahringer, **Bi-criteria Scheduling of Scientific Workfows for the Grid**, Eighth IEEE International Symposium on Cluster Computing and the Grid, pages 9-16, 2008.

[172] V. T'kindt, J. C. Billaut, **Multicriteria Scheduling. theory, models and algorithms**, Springer, 2nd Edition, 2006.

[173] H. Topcuoglu, S. Hariri and M. Y. Wu, **Performance effective and low-complexity task scheduling for heterogeneous computing**, IEEE Trans. on Parallel and Distributed Systems, Volume 13, Issue 3, pp. 260-274, 2002.

[174] E. Ilavarasan and P. Thambidurai, **Low complexity performance effective task scheduling algorithm for heterogeneous computing environments**, Journal of Computer Sciences, Volume 3, Issue 2, pages 94-103, 2007.

[175] D. Breitgand, A. Marashini and J. Tordsson, **Policy-driven service placement optimization in federated clouds.**, Technical report, IBM Haifa Labs, 2011.

[176] R. V. den Bossche, K. Vanmechelen and J. Broeckhove, **Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads**, In Proceedings of the 3rd IEEE international conference on cloud computing, pages 228-235, 2010.

[177] R. Van den Bossche, K. Vanmechelen and J. Broeckhove, **Cost-Efficient Scheduling Heuristics for Deadline Constrained Workloads on Hybrid Clouds**, on Cloud Computing Technology and Science, IEEE International Conference, pages 320-327, 2011.

[178] R. Buyya, C. S. Yeo, and S. Venugopal, **Market-Oriented Cloud Computing : Vision, Hype, and Reality for Delivering IT Services as Computing Utilities**, Proc. Of the 10th IEEE International Conference on High Performance Computing and Communications, pages 5-13, 2008.

Résumé

Bien que les nombreux avantages d'utiliser des ressources hébergées dans le *Cloud* soient avérés, celui-ci reste confronté à de nombreux problèmes qui peuvent compromettre son succès commercial. Parmi ceux-ci, on peut citer le manque d'aide aux utilisateurs pour leur permettre de choisir au mieux les ressources disponibles. Il est donc indispensable de développer des méthodes nouvelles pour une gestion optimale des ressources dans le *Cloud*.

Cette thèse se situe dans le cadre de développement de méthodes d'optimisation d'utilisation des ressources pour l'exécution des processus dans le cadre *Cloud*. Nous avons abordé ce sujet sous trois aspects complémentaires prenant en compte des critères de qualité de service conflictuels.

Pour le premier aspect, nous avons proposé un ensemble de stratégies d'allocation de ressources et d'ordonnancement de tâches d'un processus scientifique dont les tâches sont supposées automatisées.

Pour le deuxième aspect, nous avons proposé une extension des premières stratégies pour prendre en compte le fait que certaines tâches nécessitent l'intervention d'une(des) ressource(s) humaine(s) pour s'accomplir.

Pour le troisième aspect, des méthodes d'optimisation afin d'assurer un accès équitable à des ressources, partagées par un ensemble d'instances d'un même processus métier, ont été proposées.

Mots-clés: Processus métiers et scientifiques, *Cloud Computing*, Optimisation multi-objectifs, Allocation de ressources et ordonnancement.

Abstract

Despite the many proven benefits of using resources deployed in the cloud, it still faces many problems that can compromise its commercial success. Among theses problems, we can mention the lack of methods allowing users to choose the best available resources. Thus the development of new methods for optimal resources management is necessary.

The aim of this PhD thesis is based on this topic by the development of methods to optimize the use of resources in the *Cloud* context. To achieve this goal, our approach is driven by three complementary aspects taking into account conflicting quality of service criteria.

For the first aspect, we have proposed a set of resources allocation and tasks scheduling strategies for a scientific process consisting of automated tasks.

For the second aspect, we have proposed an extension of these strategies to take into account the fact that some tasks require the human resource(s) intervention.

For the third aspect, optimization methods in order to allow a fair access to the shared resources by a set of business process instances are proposed.

Keywords: Business and scientific processes, *Cloud Computing*, Multi-objective optimisation, Resources management and scheduling.